HP 3000 IUG EDINBURGH

2nd-7th OCTOBER 1983

HP 3000 INTERNATIONAL USERS GROUP

1983 HP 3000 INTERNATIONAL CONFERENCE
EDINBURGH OCTOBER 2–7
THE ASSEMBLY ROOMS AND
MUSIC HALL COMPLEX, GEORGE STREET
EDINBURGH, SCOTLAND

PROCEEDINGS

Host Committee Members

Tim Cullis
Ivor Davies
John Eaton
Jean-Pierre Gueguen
Stuart Jones
Steve Mathews

IUG Staff

William Crow, Association Manager
Aldena Duval, Membership Services Manager
Renaye Lee, Conference Manager
Ben Lloyd, Technical Manager
Leslie Nicholson, International Liaison Manager
Nancy Trankle, Membership Activities Manager

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

LIST of PRESENTATIONS - by TITLE

Computer Museum

Anticipating OSI
by Bjorn Vermo 37-1

The Architecture of Integrated Office Systems
by Peter Williams 60-1

Architectural Changes for MPE V
by David N. Holinstat 70-1

Automating Systems Development with a Data Dictionary
by David C. Dummer 13-1

Capital Project Control Systems
by David Harvey 51-1

A Case Study on the Installation of Manufacturing Systems
by R.J. Woodhead, B.Sc. MBPICS 52-1

A Communications Package for Internetworking on the HP 3000
by Reinhold Leitner 30-1

Companies Statutory Books on the HP 3000
by N. Bedford 24-1

A Comparative Study of RAPID and COBOL
by John Sanders 21-1

A Comparison of The OSI Reference Model & Hewlett-Packard's
DS3000 Telecommunications Software
by Gregory J. Sannik 27-1

The Computer User/Network Interface at Glaxo Pharmaceuticals Limited
by Glaxo 29-1

Computer Megatrends of the 80's Affecting Business Productivity
by Barry Klaas 45-1

Computerized Personnel Systems
by Tony Ive 23-1

Criteria for Selecting Software Packages for the HP 3000
by Thomas A. Wilson 41-1

Customizable Software - Why You Need It
by Mike Kolsolchroen 11-1

The Data Dictionary: An Emerging System Resource
by Alan T. Pare 10-1

Data Bases on Micro-Computers. A Reality.
by Michel Kohon 78-1

Data Communications for the HP 3000 User
by Micom 39-1

Data Dictionaries That Cost Nothing
by Robert A. Evans 14-1

Data Model Financial Modeling for the HP 3000
by Edward Humphrey 56-12

Database Techniques - IMAGE Versus KSAM
by Dipl. Ing. Jorg Grossler 82-1

Decentralizing the DP Function
by Nicholas H. Cuthbert 46-1

Developing Large Integrated Systems Using RAPID/3000
by M.P. Ashdown 18-1

Development of the Britoil Integrated Terminal Network
by N.H. Shelness 35-1

Direct Screen Addressing Using COBOL II
by J. Birkhead 15-1

Distributed Processing in an IBM-HP Environment
by Rolf Frydenberg 28-1

Effective Site Planning Strategy
by Lynn K. Darnton 73-1

Electronic Newsroom System
by Systemsolve 58-1

The FCS-EPS Decision Support System - Business Planning on the HP 3000 Computers
by K.F. Ltd 62-1

For the Mis Manager: Even If Life Gives You Lemons, You Can Make Lemonade!
by Margaret T. Van Vliet 43-1

Getting Started in Data Capture
by Bruce Toback 49-1

Grampian Software Facilities Ltd. H-PAY
by 25-1

Hoskyns Integrated Purchasing System
by Nigel Fielden 48-1

How HP Networks Can Improve Your Productivity
by Jay Kidd 34-1

How to Avoid Saying ":Hello" - Running Online Applications on the HP 3000 Without Access to MPE
by John Parkinson 76-1

How to Get a High Performance Order-File
by Hanne Hansen, M.Sc 7-1

IAS/3000 Integrated Accounting Systems For the HP 3000
by 22-1

IMAGE Design: Documentation of Structured Analysis Techniques Using Dictionary/3000
by Richard Irwin 1-1

IMAGE Design: Logical Data Base Mapping
by Timothy Cullis and Richard Irwin 3-1

IMAGE Design: Structured prototyping
by Timothy Cullis 2-1

IMAGE and ADAGER: The Dynamic Duo.
by F. Alfredo Rego & Fred White 81-1

IMAGE/3000 Strategy
by Wendi Brubaker 9-1

ITA/3000 A User-Friendly Screen Form Handler Operating in Character Mode
by Kurt Sager 80-1

Implementing a Manufacturing System in an Engineering Company
by Stephen M. Foster, B.A., M.A. 47-1

Improving Database Application Performance Without Changing Your Programs
by 20-1

Improving Hardcopy Output for Better Efficiency
by Matt Cuson 66-1

Integrated Systems for Manufacturing Companies
by P. Robinson 53-1

Interprocess Communication in a Network Environment
by Nick Bates and Gary Wolstenholme 36-1

A KSAM Handler
by Bruno H. Freudenthaler 4-1

L'Air Liquide
by D.S.I.O./S.E.I. JPB/GO 12-1

Linguistic Aspects of Word Processing
by Franz-Josef Boll 56-1

MPE V: Product Overview, Project Development Strategy, and Implementation Methodology
by Robert L. Mead 72-1

MPE in an Oil Industry Environment
by J. Moran and McTurnill 67-1

MPE-Disc Cache in Perspective
by John R. Busch and Alan J. Kondoff 69-1

Making Sense of Office Automation
by Trevor Wing 59-1

Making Utilities Non-utilitarian
by Chris Shinn 74-1

Networking Made Simple
by Jim Geers 32-1

New Product Announcement from Direct
by Bernard Lovell 65-1

Nucleus: A System Management Tool
by W. Gary Sitton 79-1

One-Line Support: An Inside Look or the Buck Stops Here (almost)
by Karen Biglarderi and Nancy Ofslager 68-1

Optimizing System Performance
by David S. Wertheim 69-9

Personal Computers in Networked Systems
by Bruce Woolpert 57-1

Programming for Performance
by Jim May 19-1

The Pains & Pleasures of Office Automation the HP Way -
A Case History of Office Automation at Glaxo Pharmaceuticals
by M.H. Wadsworth 55-1

"PERSON - The Personnel System That's Different"
by 26-1

Planning and Implementing a Corporate Data Center
in an International Environment
by Norman W. Davis 42-1

RELATE/3000 - Relational DBMS for the HP 3000
by Assyst 77-1

Remote Line-Printing
by Michael P. Mansfield 63-1

Remote Spooled Printers in an Operatorless Environment
by Tor Kristian Hande 71-1

Simplify Workstation to Host Connection
by Hewlett-Packard 40-1

Software Prototyping: Today's Approach to Information Systems
Design and Development
by Orland Larson 8-1

Solve Your Problems with Imagination
by Erik Wendelboe 75-1

Some Preliminary Suggestions For Transact Programming Standards
by Tony Seymour and Lawrence McNamara 16-1

The State of Affairs with Hewlett-Packard's X.25 Product Offering
by Hewlett-Packard 38-1

Structural Engineering on the HP 3000 - Creating an Environment for the Engineer
by B. Brennan and G. Stewart 50-1

Synchronous Communication on the HP 3000
by N.M. Demos 31-1

Systems Development, Projects Life - The Practical Experience
by Carl Cristian Lassen, M.Sc. 44-1

Technical Publication Costs Cut in Half with Laser Printing
by Steve Wilk and Sam Boles 64-1

Techniques of Local Area Networking
by Marc Burch 33-1

Triple Convergence; Scottish Library Information Systems on the HP 3000
by Bruce Royan 5-1

Tingler or What sort of life do programs lead at run-time?
by Lance Carnes 83-1

A Tree Grows in IMAGE: Data Structures in the IMAGE Database
by Theodore Dillenkofer, Jr. 6-1

UNIX - An Introduction
by M.J. Bailey and John O'Leary 17-1

User Training in Office Systems
by Jay Young 61-1

Using Plotters in Word Processing
by Rolf Frydenberg 54-1

Database Consultants europe b.v.
Keizersgracht 557, 1017 DR Amsterdam. Telefoon +31 20 - 22 42 43

IMAGE DESIGN: Documentation of structured analysis techniques using DICTIONARY/3000

A presentation to the
HP3000 International Users Group
European 1983 Conference
Edinburgh October 3rd - 6th

Richard Irwin

Database Consultants Europe
Keizersgracht 557
1017 DR Amsterdam
The Netherlands
Tel: (020) 22 42 43

Abstract

This is the first in a trilogy of papers given jointly by Tim Cullis and Richard Irwin addressing the subject of data base design using structured methods.

The first paper concentrates on the twin subjects of Data Analysis and Activity Analysis. It investigates the concepts of modelling the data area and business activities as both an analysis method and a communications medium with the user.

Methods are also introduced of using DICTIONARY/3000 as an integral part of the initial analysis documentation.

The other two papers in the trilogy cover **Structured Prototyping** and the **Logical and Physical Design of IMAGE** file structures. The three papers are related; in the presentation of our papers, we assume attendance of previous sessions.

1. INTRODUCTION

Ever since the early commercial computer systems we have been striving to find a scientific yet understandable method for the analysis and design phase of a project. Even within the construction or programming phase it is still clear that there are no hard and fast rules that can be applied to guarantee a satisfactory result. One of the major reasons for this is because a computer system is very personal to any one particular company or department and yet we still continue to hide away in our ivory towers, known as data processing departments, developing computer systems that we think the user would like. It is very difficult for a user to adapt to something new when he is most likely short of time in any case. This problem is only compounded when the user cannot identify, in his terms, any tangible benefit. The user must feel motivated and therefore involved in each stage of his system. The objective of this paper is to demonstrate a method widely used by our clients to ensure understanding of a project by all members of a team including the user. This method is independant of any particular hardware/software but in this paper the standard tools of Hewlett Packard are used to demonstrate its applicability within projects destined to be developed on the HP3000.

2. SCOPE

The method embraces a project from conception to final implementation as shown in figure 1. It is interesting to look at the background to the method at this stage. With the introduction of on-line systems it became clear that the data structure within computer systems needed careful thought as different processes were now working off the same data from all different views. This meant that the data had to have its own meaning without the support of any particular application program. Hence, the introduction of database management systems (DBMS) software that could look after the data without those application programs. Data dictionaries have also been introduced to centrally define the data. It is expected that in the very near future data dictionaries/directories (DD/D) and DBMS's will be merged to produce a piece of software to look after the definition/location and integrity of our data. With these powerful facilities some method was needed to construct the correct data structure and suitable application

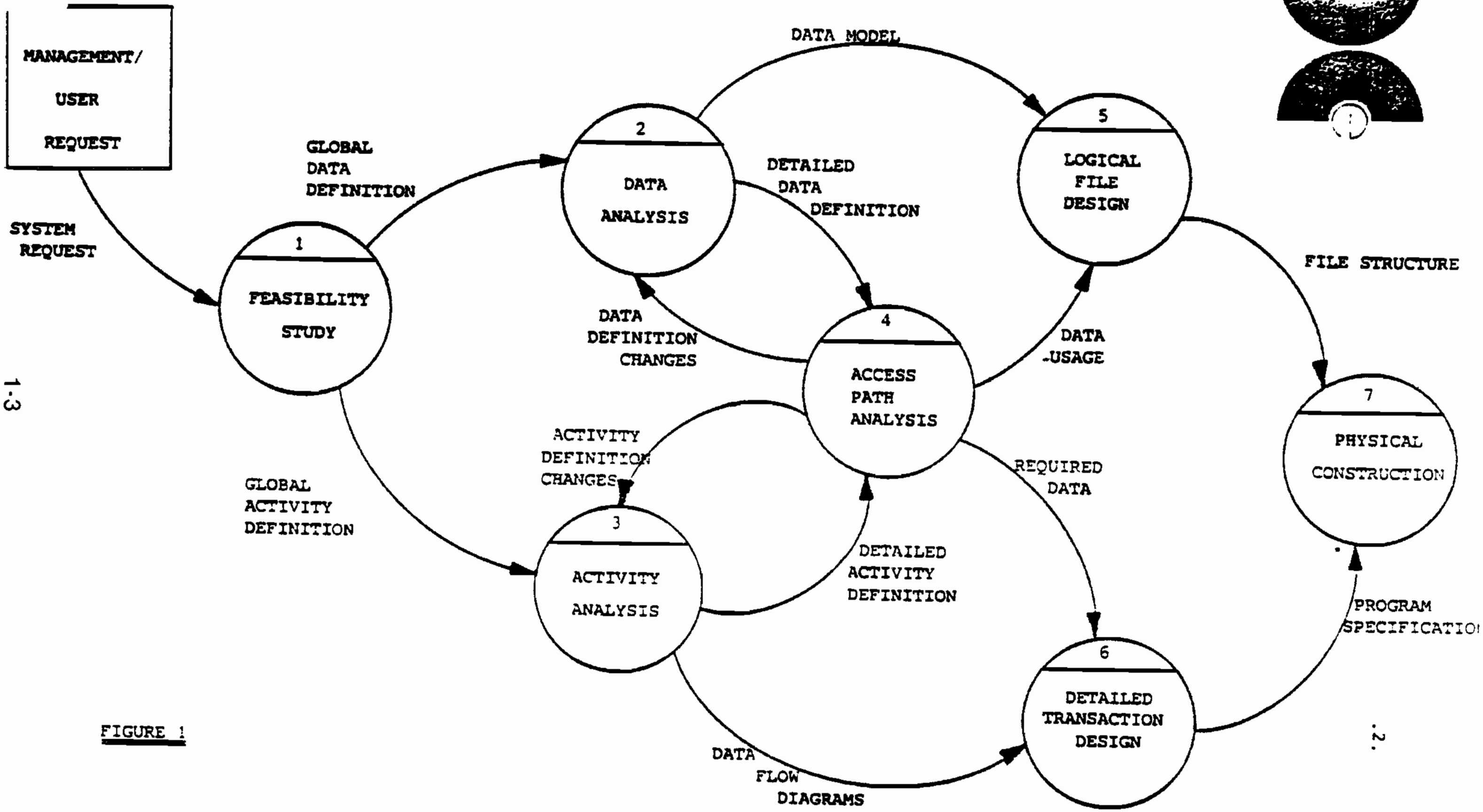

MANAGEMENT/
USER
REQUEST
SYSTEM REQUEST
1
FEASIBILITY STUDY
GLOBAL DATA DEFINITION
GLOBAL ACTIVITY DEFINITION
2
DATA ANALYSIS
DATA MODEL
DETAILED DATA DEFINITION
DATA DEFINITION CHANGES
3
ACTIVITY ANALYSIS
ACTIVITY DEFINITION CHANGES
DETAILED ACTIVITY DEFINITION
DATA FLOW DIAGRAMS
4
ACCESS PATH ANALYSIS
DATA USAGE
REQUIRED DATA
5
LOGICAL FILE DESIGN
FILE STRUCTURE
6
DETAILED TRANSACTION DESIGN
PROGRAM SPECIFICATIO
7
PHYSICAL CONSTRUCTION

FIGURE 1

programs that would work. This is why this method emphasizes the split between data and activity analysis at the earliest possible point in the project.

An interesting development has taken place since the introduction of database, all kinds of development tools and utilities have blossomed forth now that data can be centrally defined and therefore generalised routines can be written to work effectively. As a result even the simplest of applications are now adopting a database approach in order to obtain the payoff from these software products. Many people have come unstuck at this stage because they did not realise the basic requirement for these new tools was to define the data properly. The method used in this paper can be applied to most, if not all, commercial computer projects. The activities of data analysis and activity analysis are covered in this paper, the subject of logical file design being covered in a separate paper. DICTIONARY/3000 has been taken as the dictionary tool. An example project has been taken to demonstrate a practical example of the method at work. This example is an actual system called TRAP/3000, Time Reporting And Planning system.

3. DICTIONARY/3000

Before continuing any further a closer look at DICTIONARY is necessary. Figure 2 shows a global data model of the DICTIONARY/3000 itself. It can be seen from the diagram that the dictionary can be split into two major areas

- ACTIVE	meaning that data entered in this part of the dictionary can be used by an automated process to build a database, program data structure (in the case of TRANSACT) or an enquiry.
- PASSIVE	meaning that this part of the dictionary is only used for documentation and will not influence any automated process.

It will now be demonstrated how the passive part of the dictionary can be made use of, right from system conception through to implementation.

CATEGORY will be used to categorise data areas, subsystems and entities. (entities will be defined shortly)

DICTIONARY / 3000 GLOBAL DATA MODEL

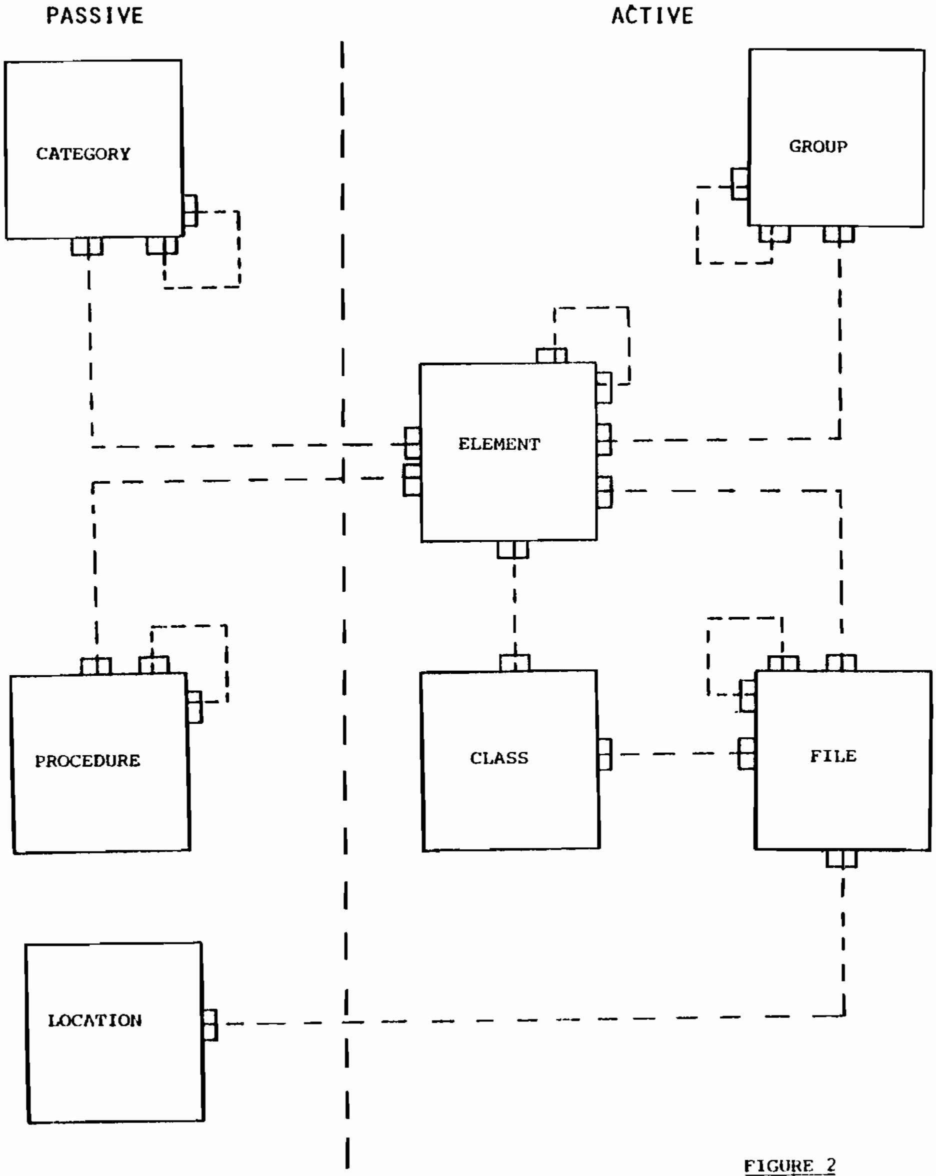

PASSIVE
ACTIVE
CATEGORY
GROUP
ELEMENT
PROCEDURE
CLASS
FILE
LOCATION

FIGURE 2

ELEMENT will be used to uniquely identify attributes of entities, data sets being passed between activities within data flows and any derived fields within activities.

PROCEDURE will be used to define any level of activity from a business area right down to a single program definition.

4. FEASIBILITY STUDY

The feasibility study of the TRAP system was a relatively short process. A client required automation of a manual planning and time reporting system. The usual PERT systems were examined and found not to be suitable and so a general outline of the system was built. Here is an example of some of the data areas that were identified

PROJECT - A project was defined as being something that was carried out to produce a product within a quantifiable time by the company on behalf of the same company or another company.

PERSON - A person would only be a part of the system if, he/she was carrying out a project activity.

DEPARTMENT - A department is a part of a company which the cost of a person can be attributed to.

COMPANY - A company is either a company that is responsible for paying for the project and/or supplying the people for that project.

In a small system such as this the data area definitions will be written in the report and would most likely not require any further changes. It is therefore unlikely to give much benefit to update the dictionary at this time. The dictionary will therefore not be needed until the next phase. i.e. data analysis. An example of the global activities involved in this system are as follows

PROJECT MANAGEMENT : This is the process of updating planning data to identify who does which task and when.

ORGANISATION MANAGEMENT : It is necessary to relate people to their correct departments and projects to the correct companies for re-charging purposes.

TIME REPORTING : This is the function of recording time against tasks or projects and non-project activities.

From this type of information the data analysis phase was started.

5. DATA ANALYSIS

Before describing data analysis for TRAP, it is useful to examine what data analysis is. Data analysis is a method used to understand and document a company environment in terms of its data resources. The results of data analysis are summarised in a diagram known as a data model. This is our major communication tool for user and projects team. Detailed results are documented in a structured form using the data dictionary. In order to carry out data analysis it is necessary to have direct discussions with the user to extract information from his personal knowledge, look at his manual/computer records or files and any correspondence going in/out of the department/function. The framework of data analysis consists of

ENTITIES

An entity is something of fundamental importance to a company. It is thus something about which data will probably be kept in an information handling system. e.g. Objects, people, places or abstractions such as events.

ATTRIBUTES

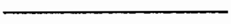
An attribute is a basic unit of information which describe an entity. An attribute cannot usefully be subdivided into other units of information. An entity must have attributes if it is of interest to the company. e.g. a code, data, quantity etc.

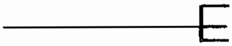
RELATIONSHIPS

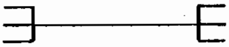
A relationship is an association between entities. It has two directions e.g. a car is made from many parts and a part exists in zero, one or more cars.

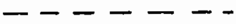
The data model is often referred to as the entity/relationship model. The rules are very simple. A rectangle is an entity with the name of the entity inside the rectangle.

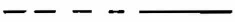
A line connecting either two entities or an entity to itself is a relationship. There are various types of relationships.

Mandatory one to one (1:1) = 

Mandatory one to many (1:n) = 

Mandatory many to many (m:n) = 

Optional relationships = 

Partially optional = 

TRAP data model can now be examined. The first pass is shown in figure 3.

From this model discussions can begin as to the suitability of the data structure. The first data model on a project is rarely correct. If there are no changes it often means that the user does not understand but assumes that you do! This is a more dangerous situation than the complete first data model being wrong. In the TRAP data model it became obvious to the user that one could not control a situation where many people were working on many projects without defining something in between. On this entity in between one could store information about the actual work carried out by one person on one project. The data model developed in this manner is shown in figure 4. One of the most important points to note is the elimination of the many to many relationships as it is unlikely that you will be able to support these satisfactorily within an information handling system. The dictionary could now be used to enter the definitions of the entities on the data model.

FIGURE 3 FIRST PASS DATA MODEL TRAP/3000

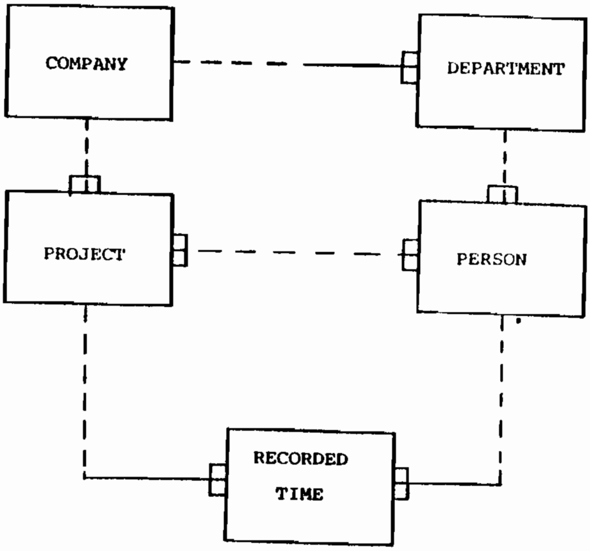
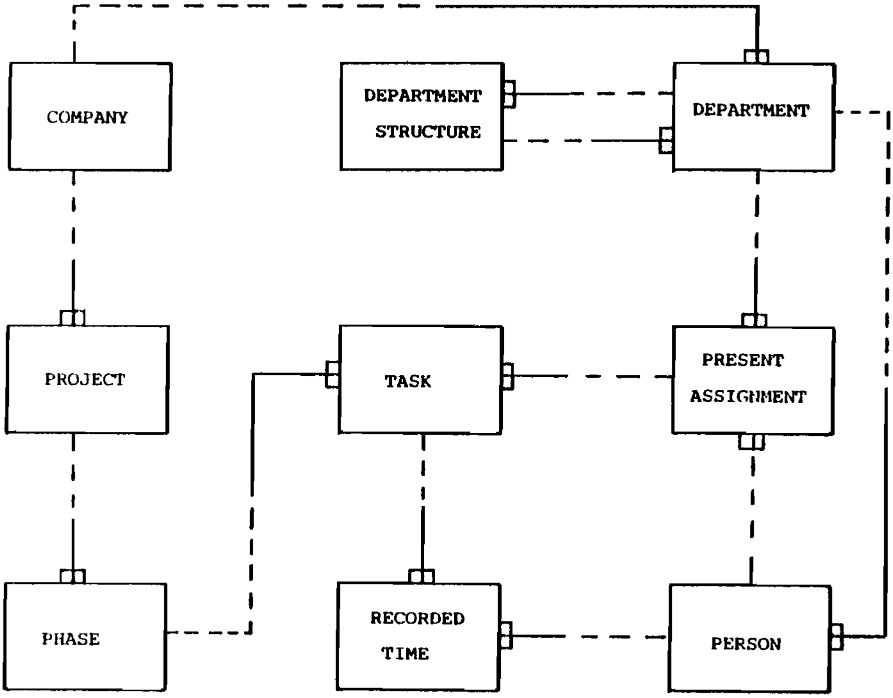

COMPANY
DEPARTMENT
PROJECT
PERSON
RECORDED TIME

FIGURE 4 Nth PASS DATA MODEL TRAP/3000


COMPANY
DEPARTMENT STRUCTURE
DEPARTMENT
PROJECT
TASK
PRESENT ASSIGNMENT
PHASE
RECORDED TIME
PERSON

DICTDBM was used with the CREATE CATEGORY command e.g.

```
     CATEGORY          >   DEPT
     LONG NAME         >   DEPARTMENT
     TYPE              >   ENT (for ENTITY)
     DESCRIPTION       >   A department is a part of
                       >   a company which the cost
                       >   of a person can be attributed
                       >   to.
```

A CATEGORY was also set up with the TYPE = SDB (meaning subject database) for TRAP itself so that all entities within the TRAP system could be grouped under it. e.g.

```
     CATEGORY          >   TRAP
     LONG NAME         >   TIME REPORTING AND PLANNING SYSTEM
     TYPE              >   SDB
     DESCRIPTION       >   (A generalised description of
                       >    TRAP/3000 was entered)
```

The RELATE CATEGORY command was then used to associate the entities to TRAP. e.g.

```
     PARENT CATEGORY   >   TRAP
     CHILD CATEGORY    >   DEPT
     DESCRIPTION       >   (not used)
```

REPORT/3000 was used to point all entities within TRAP with their corresponding descriptions and given to the user for approval. This process went on until the team and user were happy with the definitions.

Each entity was then taken and a list of attributes constructed against it. e.g.

```
Entity :  DEPARTMENT

Attributes :   Department code (Unique identification)
               Department Name
               Company code (Key of company)
               Department entity creation date
               Date department entity last changed
               User who changed entity (Key of system user)
```

Once again DICTDBM was used to enter the definition in the dictionary as follows with CREATE ELEMENT command.

```
ELEMENT        > DEPT-CODE

LONG NAME      > DEPARTMENT CODE

TYPE           > X

SIZE           > 8

   .

   .

   .

DESCRIPTION    > The department code uniquely
                 identifies a department to
                 the system
```

The ADD CATEGORY command could then be used to associate attributes (ELEMENTS) to entities (CATEGORIES).

```
CATEGORY       > DEPT

ELEMENT        > DEPT-CODE

ELEMENT-ALIAS  > UI  (This was a trick to show
                      which attributes were
                      unique identifiers or
                      foreign keys)

DESCRIPTION    > Not used
```

With REPORT/3000 another series of outputs were used to check if the data area had been covered properly. The important ones were an alphabetical list of attributes with a full description and an entity/attribute matrix. e.g.

ENTITY : DEPARTMENT

Attribute	Keys	Type	Size
DEPT-CODE	UI	X	8
DEPT-NAME	-	X	50
COMPANY-CODE	- FK	X	8
DEP-CRE-DTE	-	X	6
DEP-UPD-DTE	-	X	6
USE-CODE	- FK	X	8

ENTITY : PERSON

.
.
.
.

etc.

With a good definition of the data at hand the activity analysis could begin.

6. ACTIVITY ANALYSIS

Activity analysis is a method used to understand and document a company environment in terms of the activities required to create, change and retrieve data (defined in data analysis). Dataflow diagrams are used at the high level to first identify the flow of data through the organisation so that the activities working with that data can be identified.

Dataflow diagrams consist of the following parts :

ACTIVITY

An activity is shown by a circle with the name inside the circle. A unique number (or letter) is placed in the top of the circle to show (if it is a sub activity) where it was derived from.

DATA SET

A data set is shown by a line between two activities. It has direction (either 'IN' or 'OUT') which is shown by an arrow on the inbound activity.

HOLDING POINT

A holding point is shown as a file symbol. This is an area where data is held for later use. It need not be a computer file.

EXTERNAL ACTIVITY

Activities that take place outside the system but help the understanding of its interaction with other systems which are identified by rectangles.

The results of activity analysis are documented in the dictionary which once again gives a central definition of activity, data set, holding point and external activity. A first pass dataflow diagram of TRAP looked like the diagram in figure 5. Each activity could be broken down into sub activities. An example is given in figure 6.

FIGURE 5 FIRST PASS DFD FOR TRAP/3000

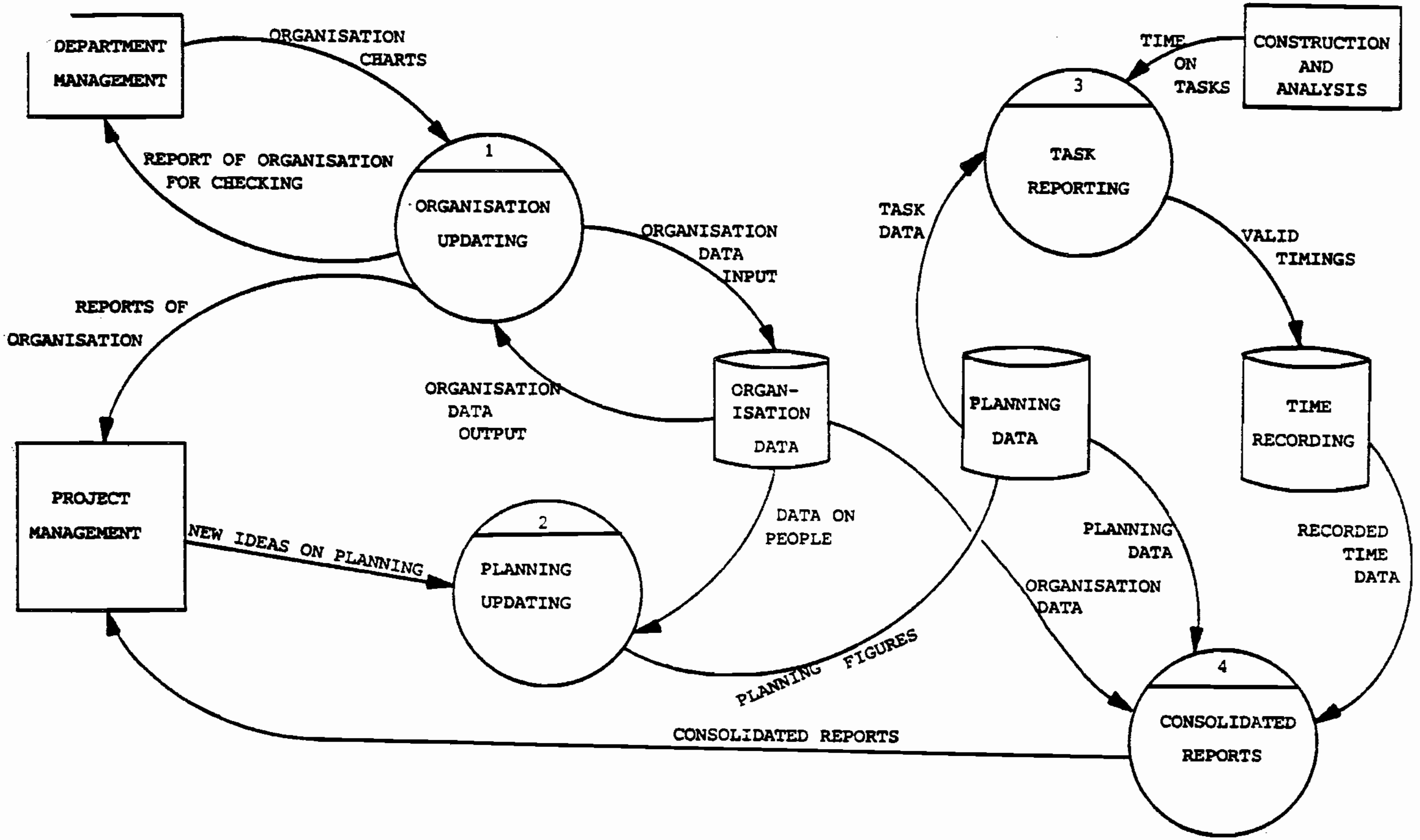
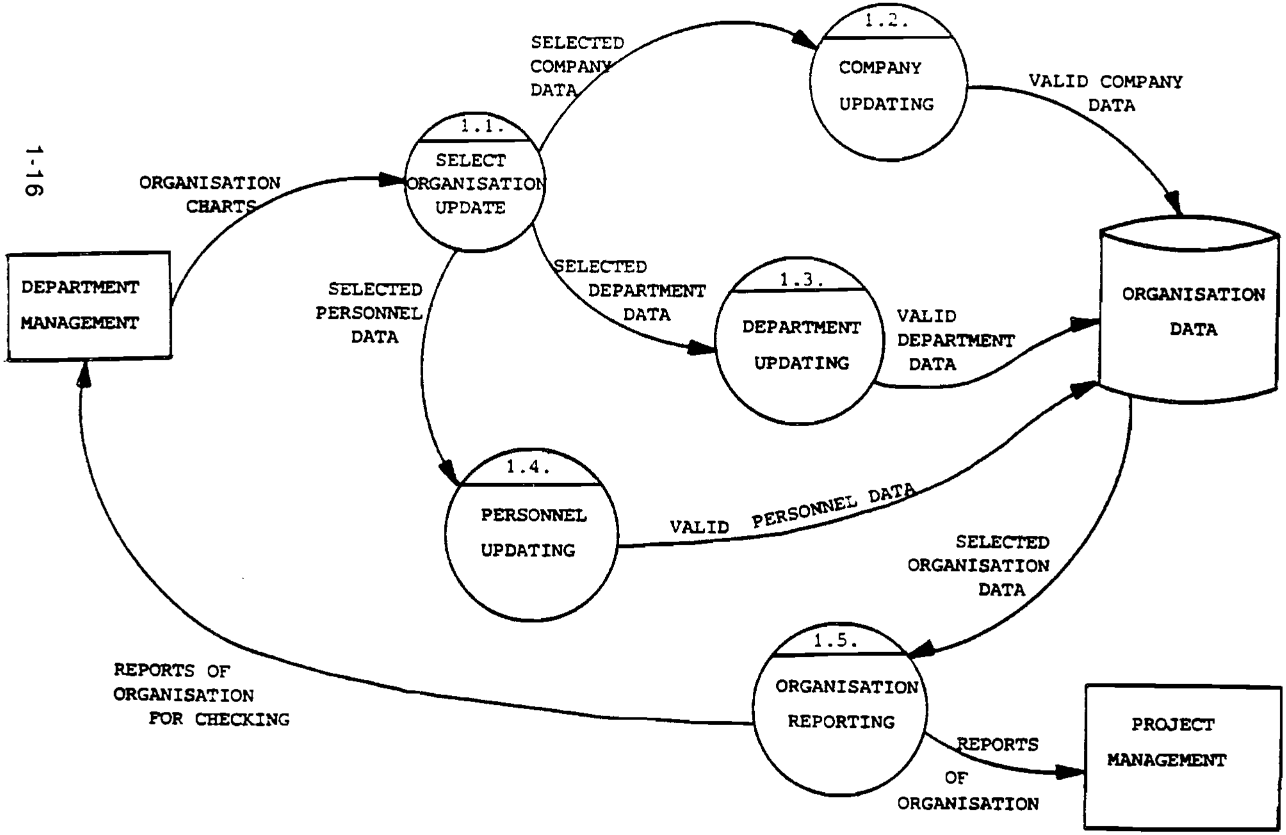

DEPARTMENT MANAGEMENT
ORGANISATION CHARTS
REPORT OF ORGANISATION FOR CHECKING
1
ORGANISATION UPDATING
ORGANISATION DATA INPUT
ORGANISATION DATA OUTPUT
REPORTS OF ORGANISATION
ORGAN-ISATION DATA
PROJECT MANAGEMENT
NEW IDEAS ON PLANNING
2
PLANNING UPDATING
DATA ON PEOPLE
PLANNING FIGURES
CONSOLIDATED REPORTS
CONSTRUCTION AND ANALYSIS
TIME ON TASKS
3
TASK REPORTING
TASK DATA
VALID TIMINGS
PLANNING DATA
TIME RECORDING
PLANNING DATA
ORGANISATION DATA
RECORDED TIME DATA
4
CONSOLIDATED REPORTS

FIGURE 6 ACTIVITY ANALYSIS TRAP/3000

ORGANISATION UPDATING


DEPARTMENT MANAGEMENT
ORGANISATION CHARTS
1.1. SELECT ORGANISATION UPDATE
SELECTED COMPANY DATA
1.2. COMPANY UPDATING
VALID COMPANY DATA
ORGANISATION DATA
SELECTED DEPARTMENT DATA
1.3. DEPARTMENT UPDATING
VALID DEPARTMENT DATA
SELECTED PERSONNEL DATA
1.4. PERSONNEL UPDATING
VALID PERSONNEL DATA
SELECTED ORGANISATION DATA
1.5. ORGANISATION REPORTING
REPORTS OF ORGANISATION
PROJECT MANAGEMENT
REPORTS OF ORGANISATION FOR CHECKING

The dictionary was then updated using the PROCEDURE part as follows :

```
REPEAT CREATE PROCEDURE

PROCEDURE    >   TRAP
LONG NAME    >   TIME REPORTING AND PLANNING SYSTEM
TYPE         >   ACT (Short for ACTIVITY)
LANGUAGE     >   Not used
DESCRIPTION  >   (Brief overview of the TRAP
                  activities)
PROCEDURE    >   SELECT-ORG-UPD
LONG NAME    >   SELECT ORGANISATION UPDATE
TYPE         >   ACT
LANGUAGE     >   Not used
DESCRIPTION  >   (Description of the activity)


      .
      .
      .

etc.
```

The RELATE PROCEDURE command was then used to build the activity 'structure' as follows :

```
REPEAT RELATE PROCEDURE

PARENT PROCEDURE    >   TRAP
CHILD PROCEDURE     >   ORG-UPD
DESCRIPTION         >   (Not used)
      .
      .
      .
      .
```

PARENT PROCEDURE > ORG-UPD
CHILD PROCEDURE > SELECT-ORG-UPD
DESCRIPTION > (Not used)

External activities and holding points were also defined with the types 'XACT' and 'HOLD' respectively.

Data sets were created as ELEMENTS and the ADD PROCEDURE command was used to associate the dataset to a particular activity and show its direction as follows :

PROCEDURE > SELECT-ORG-UPD
ELEMENT > ORG-CHARTS
ELEMENT-ALIAS > IN (This is a trick to show the direction of the data set)
DESCRIPTION > Not used

A report was then constructed to show the definition of activities, external activities and data sets.

Sub activities finally become a transaction process within a system e.g. DEPARTMENT UPDATING is actually a program that creates and updates the DEPARTMENT and DEPARTMENT STRUCTURE in the database. At this stage it is useful to identify the attributes belonging to the dataset. These attributes should of course, already exist as a result of the data analysis. That is nice in theory but in practice you very often have to re-examine the data analysis as a result that your findings in the activity analysis. The attributes belonging to a data set were associated as follows using the RELATE ELEMENT command.

PARENT ELEMENT > SELECT-DEPT-DATA
CHILD ELEMENT > DEPT-CODE
.
.
.
.

etc.

A report of where each attribute was being used and/or changed was then produced to check that all data within the system was being used in some way. From this information, access path analysis could now begin (not covered in this paper).

7. CONCLUSION

Once everything is documented in the dictionary it is a very simple task to keep it maintained. Any new personnel coming onto the project have an easy job when wishing to find out the stage of the project and what it is all about. When using IMAGE/3000 the attributes can mostly be used directly as data items (with a few compromises for compound keys) once again providing an easy method of maintenance. It is always a nice feeling when you implement that new system and you have the confidence that the user knows what the system is going to give him and your documentation is also up to date!

Database Consultants europe b.v.
Keizersgracht 557, 1017 DR Amsterdam. Telefoon +31 20 - 22 42 43

IMAGE DESIGN: Structured prototyping

A presentation to the
HP3000 International Users Group
European 1983 Conference
Edinburgh October 3rd - 6th

Timothy R. Cullis

Database Consultants Europe
69, Grasmere Gardens
Harrow Weald, Middx.
HA3 7PS England
Tel: (01) 863 2428

Abstract

At the 1982 European Copenhagen conference, prior to my joining DCE, Richard Irwin and I discussed putting forward a trilogy of papers for the Edinburgh meeting. The intention was to reflect our joint thoughts regarding what has become known as **Structured Analysis** and how structured methods may be used on HP3000 projects, especially in respect to data base design.

When analysing any situation, it is always a good idea to step back and take the 'helicopter view' of the surroundings. In order to design for shared data environments with IMAGE, it is essential to have a firm understanding of the data resource. Richard Irwin's paper in the last session concentrated on this and how Data Analysis and Activity Analysis are normally applied in a system development.

I prefer the role of the "devil's advocate" and have assumed that you will come across difficult situations in which you may think normal structured methods are not sufficient. My written paper concentrates particularly on the subject of prototyping and my thoughts on how, after all, this fits within the structured methods. My actual presentation will focus on applying prototyping concepts in your own projects.

In the next session, we will cover mapping our results to an IMAGE logical data base design. The first two papers paint the background for the final IMAGE design; the three papers together form the 'IMAGE design' trilogy.

The computer industry's track record of success (?)

It is a horrifying statistic that over half of all computer projects are either never implemented or are retrospectively judged not to have met original requirement definitions. The reasons for this are varied and examples include:

Emperor's Clothes syndrome
The company embarked on an online system which was to be implemented nationally via a multidropping multiplexing network. The raison d'etre of the new system was the over ambitious management sales targets. The "Emperor's Clothes" syndrome prevented anyone questioning clearly overstated needs.

Ten months into the project it was scrapped and replaced by a postal batch service into a centralised data control department.

Frozen requirements specification
Because the requirements specification was frozen during the two year development lead time, when the system came to be delivered it was not suitable for the revised methods of working which were adopted a year into the project development.

Modifying the system at the implementation stage doubled the project spend.

Unrealistic deadlines
The company management thought the project team were working on a 'state-of-the-art' system revision on the new computer. In reality, due to the unrealistic time constraints, the team were carrying out a rough and ready conversion from the old machine in a vain attempt to meet deadlines.

The converted system was subsequently rewritten!

Underlying reasons

The underlying reasons are a combination of two factors: lack of communication and confusion about how to proceed to tackle a problem.

Communications: How to translate user and management wishes
Goal/objective setting
Analysing for change in the organisation
General lack of standards and 'standard methods'
How to control/check/show progress?

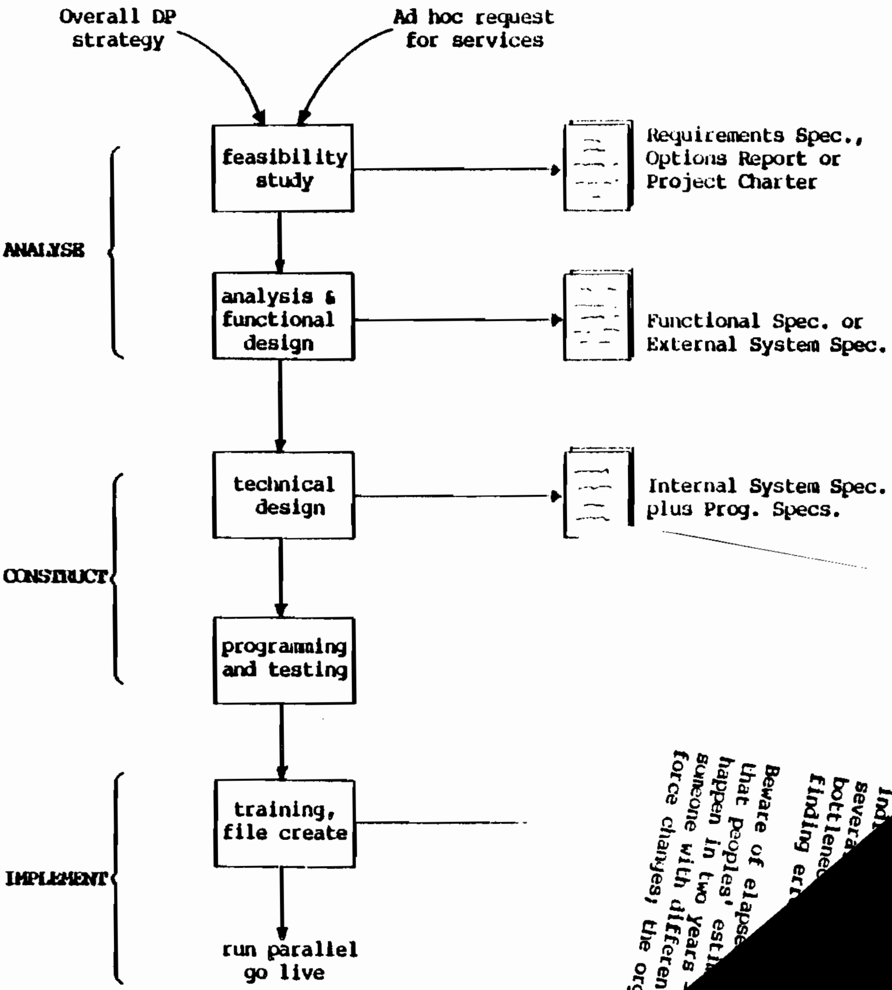
Methods: How to carry out Business Analysis
How can the users understand this complex subject?
How do we proceed from here?

Using traditional methods, it is difficult to imagine how we can improve the current track record. Let's have a look at these methods and identify the shortcomings.

Classical (traditional) systems development

The classical approach came about largely as a means to introduce contractual checkpoints for software houses and clients to agree to. It assumes we can usefully predetermine timescales and requirements and that the contract exhibit (Requirements Specification) is adequate as a communications medium.

This is seldom the case, there are normally differences of interpretation between the system builder, the system funder and the system user; also the system user's requirements alter with time.


Overall DP strategy
Ad hoc request for services
feasibility study
Requirements Spec., Options Report or Project Charter
ANALYSE
analysis & functional design
Functional Spec. or External System Spec.
technical design
Internal System Spec. plus Prog. Specs.
CONSTRUCT
programming and testing
training, file create
IMPLEMENT
run parallel go live

Classical (traditional) systems development

There is obviously a need to attempt to cost justify developments which are given the go-ahead, also possibly to chose the most cost advantageous route. The theory is fine; in practise estimates produced in early stages can be hopelessly inaccurate. Using contractual methods to tie down a fixed price with a supplier is fraught with danger due to the lack of a detailed definition of what is to be built. Also, how do you evaluate potential quality?

The classical approach is typified by a linear progression, proceeding sequentially from one task to another. This is shown in the Gant chart below. Once a task is "finished", the results are cast in stone and never re-examined - this is formally recognised by many organisations who "freeze the specification".

--- T I M E --->> (and money spent)

EVENTS

Initial study	XXXX
Functional design	XXXXX
Technical design	XXXXXXXX
Programming	XXXXXXXXXXXXXXX
System tests	XXXX
File creation	XXX
Run parallel	XXX
Go live	->->->
Document	XXXXX
User in touch?	YYYYYYYYY NO NO NO NO NO NO NO NO NO NO YYY

With this approach, it is necessary to carefully plan the project using critical path analysis methods to ensure that slippages in one area don't impact dependent linear tasks.

eople who work this way normally employ bottom up testing; first modules are
vidually tested, then program groups, then the system as a whole. This has
l disadvantages: duplication of testing, creation of a system testing
k at the end of the programming development and unnecessary delay in
ors of interpretation between users/analysts/designers/programmers.

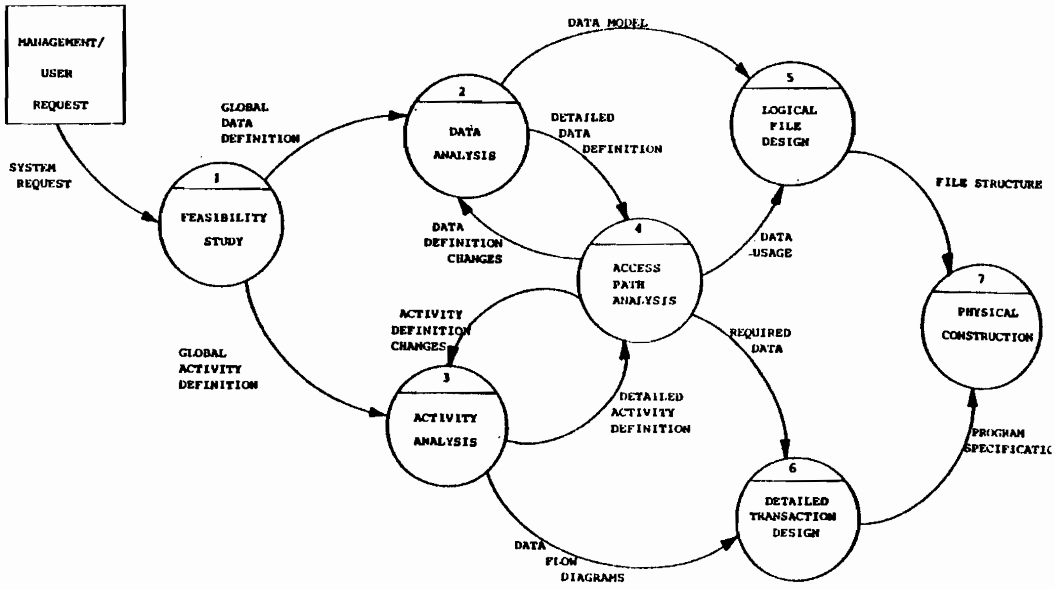
d development times in excess of one year; experience shows
ates become increasingly inaccurate with time. A lot can
the user manager could be promoted and replaced by
ideas; the system funder may retire; technology may
ation objectives may change etc.

Structured systems development

The steps in Structured Analysis are shown below in the form of a Data Flow Diagram (itself one of the methods of documenting Activity Analysis).

In the previous session, Richard Irwin explained the various stages of Structured Analysis and covered the twin subjects of Data Analysis and Activity Analysis. These are the first steps in a structured systems development.

Later in this trilogy we will deal with Access Path Analysis and the Logical and Physical Data Base Design.


MANAGEMENT/
USER
REQUEST
SYSTEM
REQUEST
1
FEASIBILITY
STUDY
GLOBAL
DATA
DEFINITION
2
DATA
ANALYSIS
DATA MODEL
5
LOGICAL
FILE
DESIGN
DETAILED
DATA
DEFINITION
DATA
DEFINITION
CHANGES
4
ACCESS
PATH
ANALYSIS
DATA
USAGE
FILE STRUCTURE
7
PHYSICAL
CONSTRUCTION
ACTIVITY
DEFINITION
CHANGES
GLOBAL
ACTIVITY
DEFINITION
3
ACTIVITY
ANALYSIS
DETAILED
ACTIVITY
DEFINITION
REQUIRED
DATA
6
DETAILED
TRANSACTION
DESIGN
PROGRAM
SPECIFICATI
DATA
FLOW
DIAGRAMS

Normally I would advocate the route of Structured Analysis and Design. There are circumstances, however, where it may be useful to incorporate prototyping to some degree.

A couple of pages on, I start to explore various methods of prototyping and where these may be useful. Before going further, however, it is useful to examine the different types of DP related systems and also some of the misapprehensions we labour under regarding system users/funders.

Different types of DP related systems

Organisations initially purchased computers to streamline administration and repetitive clerical functions such as payroll. As these were accounting orientated applications, control of DP was vested in the Accountant or VP Finance. In larger organisations, many of the easily identified business application areas have now been computerised, future developments being:

Further refine existing applications	(Routine DP)
Attempt to consolidate applications	(Information systems)
Enter new spheres	(Decision support, OA)

System Classification	Main Payoffs	Design concerns
Routine DP (eg Payroll) Predetermined tasks, decision rules and transaction flows, often high volume. Routine DP forms the bulk of conventional systems.	Business efficiency fast turnaround, quick answer to enquiries.	Efficiency of bulk operations. Matching work patterns.
Information systems (MIS) Consolidation of Routine DP systems, to answer unanticipated queries or queries which generate secondary operations.	Better information to decision makers.	Powerful end-user friendly language. Effectiveness of mgmt operations.
Decision support systems Unstructured use, not fully anticipated. Does not automate the decision but supports the executive who makes the decision.	Better decisions (eg "Head up" aircraft tactical display: kill/no kill).	The user may wish to maintain his own data base.
Process control Extremely structured tasks. Huge volumes of data, but normally only kept for short periods (gauge display).	Better and faster control of complex processes.	Speed of process (real time)
Special computational CPU dependent processes such as CAD/CAM for engineering product design, array processing for weather forecasting etc.	Speed of analysis of data, interaction with designer.	Complex technical data, normally unconnected with routine DP system
Office automation Support of secretarial services such as document production, appointment setting etc. High degree of integration in future with communications.	Initially aesthetic until sufficient communications links are in place.	Ease of use with long finger nails (eg: infra red touch screens).

Common misapprehensions regarding system users/funders/builders

I referred earlier to the overall computer industry "success rate". This stems from lack of communication and misunderstanding about the role of the user. Users are not "system builders", yet we labour under many misapprehensions:

Users know what they want: They are able to functionally describe their activities. They have a complete understanding of 'state-of-the-art' computer methods and are consequently able to decide their particular requirements. They also agree amongst themselves.

They subjectively cost analyse potential solutions: Users perceive what is expensive and what is cheap to produce. They only want cost effective solutions.

Users are wildly enthusiastic about new systems: They forgive you your previous failures, they believe you can achieve miracles. They accept that a computer solution is the right one. When the system is delivered they will accept it without criticism.

They are fully committed to the project: They have the time to fully specify their needs to you. These needs will not change during the course of development. At least one user will be allocated to the project full time to assist and co-ordinate.

Management is committed to quality solutions: You will be allowed sufficient time for investigation and analysis work before starting the implementation. After the system is delivered you will be allowed to complete the documentation in detail. You will have a 'post-implementation audit' of the system.

There are no politics involved: The EDP personnel (who report to the Accounting/Finance wing of the organisation) will be allowed total freedom to investigate Production Depts and question established work procedures etc.

Users understand Management objectives: Management has identified objectives. Change studies are carried out to establish who/what/when are involved in change in the organisation, the ramifications are understood.

The system builders are asked to develop software on time, on budget, which forms the building block of the future ...., which meets the known (and unknown!) requirements of users. It is in this environment that system builders are turning away from classical methods and looking at prototyping as the universal panacea for future development.

Let us then look at prototyping and define what is meant and what we hope to achieve by this method.

What is a prototype?

Engineering prototype

The computer industry usage of the word 'prototype' is a rather unfortunate misnomer due to the confusion with the automobile/engineering usage. The engineering connotation is that of an archetype, the pattern of perfection, 'the first or original type or model from which anything is copied'.

In this context, development of a prototype is the prelude to full scale production. The prototype exhibits the essential features and functions of the final product and probably costs millions of dollars to develop. This has no parallel with DP; our problems are one off - not mass production.

Throw away/simulation

Within our industry, one connotation of the word 'prototyping' is that of a throw away system. The system is developed in a very high level language (such as RAPID) and shows some of the proposed features; the user agrees the design and the system is then developed 'properly' in COBOL/SPL etc.

I would prefer to refer to this as a 'simulation system'. The major difference is that the simulation system is incapable of being used; whereas a prototype automobile would normally be capable of being driven on the roads.

Simulation systems may be used at the Requirements Definition or Functional Specification stages as an alternative to the boring 5000 page Victorian Novels which masquerade as decision requests reports and documentation.

Structured (top-down) prototyping

Working on the HP3000, it is most likely that you will be building Routine DP type software. This environment is typified by well defined transaction flows and predetermined tasks. Your most productive usage of prototyping will be in 'skeleton systems' and the top down development of user requirements. This is the main area which my presentation at Edinburgh will address.

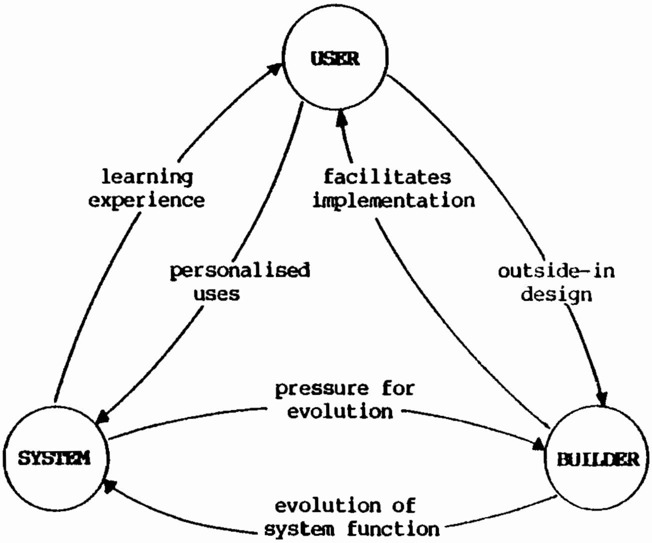
Expert system/true prototyping

The major use of full prototyping today is in the area of Decision Support Systems. By Decision Support Systems, I do not mean VISICALC!, but rather user systems which directly rather than indirectly affect decisions. This use of prototyping is best described by Keen's triangle overleaf.

Decision Support Systems

The term Decision Support System is pertinent to a situation where a complete solution can only be developed by an adaptive process of learning and evolution. It is recognised that conventional analysis routes will not supply the user feedback - perhaps the problem cannot be described until solutions are apparent.

Additional problems are introduced when working in this manner: when do you stop the iterative process of analysis, design, feedback; any system produced becomes self extinct if the iterative process is allowed to continue.


USER
learning experience
facilitates implementation
personalised uses
outside-in design
pressure for evolution
SYSTEM
BUILDER
evolution of system function

Keen's triangle above shows the three way iteration between the user, the system builder and the system itself. For example, use of the system by the user prompts change requests to the builder.

This three way learning clarifies whether true prototyping can be used as a methods of developing routine DP applications. Where there is only one 'right' way to use the system and the user must adjust to it, rather than visa versa, then this approach cannot be used in its entirety.

This is normally the case for convent ional DP projects and I would suggest you use prototyping concepts as part of a structured way of system development.

What do we hope to achieve by structured prototyping?

Prototyping is seen as a means of clarifying user needs. It has considerable benefit to the end user in that it eliminates the surprise they may have at the end of the development process - particularly annoying if their needs are incorrectly defined, left unsatisfied or implemented in an unworkable manner.

Prototyping by definition must be an interactive development process with considerable end-user involvement. Interaction of users and system builders provides for improved descriptions of user requirements; even though in the short term it **drastically reduces productivity,** leading to high initial costs.

Prototyping is not a way to avoid correct analysis of problems but it is an attempt at verifying the correctness of a design against specific problem requirements by the use of an actual construct. Speed is not of the essence; you are attempting to define detail and any software developed without due care and attention to detail is unlikely to assist in refining that detail.

There is no reason why a successful prototype cannot be enlarged/expanded when constructed correctly, into an actual working system (do not prototype in a language which is unsuited for the final system, make sure that a mix of languages could be supported). Simulation can provide much that prototyping can in the way of evaluation of solutions in a shorter time frame, but cannot be part of the final solution. Incremental system development will reduce the 'surprise' element mentioned earlier as small parts of the system come through on a regular basis; however, overall system objectives can not be evaluated until the final increment is in place.

Prototyping should be a deliberate decision as part of the development strategy and should therefore fit within the structured methods. Do not allow designers and programmers to just go off and do their own thing!

Where would you use prototyping?

In the early days of computers, many analysts had the superb background of work study/O & M. Nowadays many analysts come from an operations/programming background or straight from college, seldom have analytical skills and are really **System Designers.**

Using prototyping in this situation (ie the system builder is incapable of interpreting user wishes) may be ill advised, but is probably more suited than conventional development methods. Online interaction is particularly difficult to communicate to virgin users.

Other than 'internal sales' situations, the major use could be where the user has insufficient resources to both cope with existing work load and also assist in defining requirements. You may decide in this case to first develop a simulation system, then to further develop into a prototype and then further develop and implement.

I would, however, question management committment to the project and the likelyhood of successful implementation. One of the disadvantages of such an ad hoc approach is the de facto hardware selection.

Structured Analysis

The overall method of structured prototyping breaks into three distinct activities (assuming some form of Business Analysis or Feasibility Study has already been carried out):

Structured analysis:	breaks large complex problems into more solvable chunks using **Data Analysis** and Activity **Analysis**
Structured design	using **Iterative Prototyping** as a means of confirming decomposed structures from the analysis phase.
Structured implement:	testing on a top down basis and implementing parts of the system as applicable.

Structured analysis would be our preferred method when determining a strategic plan for an organisation, the global data and activity working papers then forming the basis of later application orientated studies. Richard Irwin's paper on data analysis describes the use of Data and Activity modelling from an individual project basis rather than in the corporate sense. Outputs from the Structured Analysis stage would include:

- Data Model showing entities and relationships
- Data Flow Diagrams showing high level and decomposed business activities
- Data Dictionary documentation including:
    - Definition of entities
    - Definition of relationships
    - Definition of attributes (at least the key or identifying attributes)
    - Where used matrix of entities/attributes
    - Definition of system procedures and external activities
    - Definition of data sets and sinks (holding points)

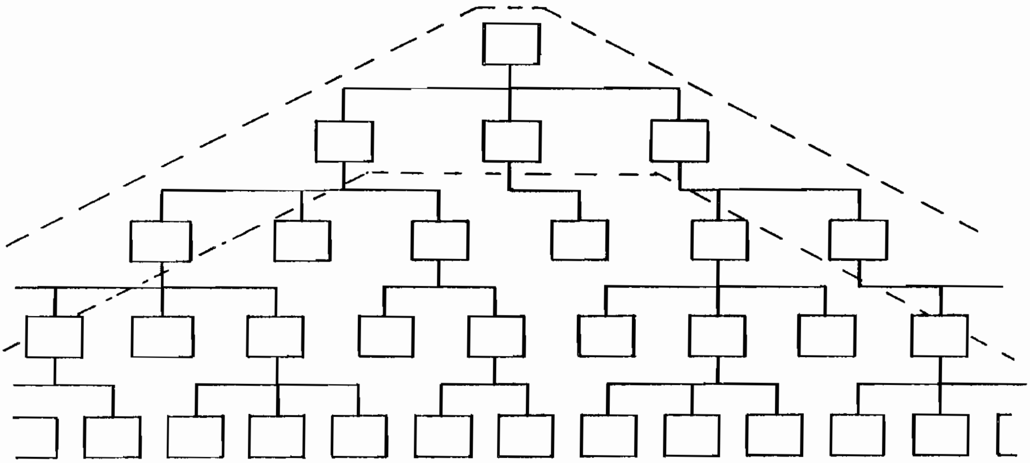
You can see from the above that the majority of analysis conclusions and documentation are derived directly from data dictionary outputs. If we were able to influence DICTIONARY/3000 development we would suggest the automation of the production of Data Models and DFDs such that ALL development information is contained in the dictionary.

> (This is not difficult - we developed a small version of this type of system for our internal use running on an Apple II with graph plotter. In many instances we are brought in to advise at the strategy planning stage, well before hardware is considered and we needed a portable system to support a data dictionary with automated graphics output for reports etc.)

Structured Analysis is an iterative process; the information we initially collect will not be complete and will be successively refined throughout the project as one's perception becomes clearer. Not all the attributes (which later map to data fields) will have been identified, probably only the important ones. It is likely that hardly any of the attributes will have type/length definitions.

Structured Prototype: first level

Having decided to use prototyping, we take the outputs from Structured Analysis and create an umbrella structure of the proposed system:



At this stage the menu screens contain a description of the lower level screens, the lower level screens probably only display the screen name and a description of the activities which would be available on the screen. It would be rather a neat convention if the screen name reflected the decomposed activity name (eg V-1.3: Dept Update).

A simple program is then used to control the simulation, allowing the user to navigate the system using function keys. Where linkages between screens are data dependent rather than fixed, this may be driven by input fields and the Enter key. By all means put some typical data fields on the forms, but do not access these or refer to them in the program other than for navigation. This type of simulation/first level prototype should be available at a very early stage in the project. In a large system, spend no more than two to four weeks defining screens and writing the driver program before showing to the user.

The feedback at this stage allows you to confirm that the overall scope of the project is correct; also that the way the functions are hung together is probably acceptable. Alteration in scope or method of system navigation can be quickly made and shown to the user.

At this stage 95% of the functionality of the program will be VPLUS screen handling and the first level prototyping could be programmed in TRANSACT. Alternatively you could develop the system in COBOL dynamics with judicious use of pre-written VPLUS/IMAGE routines. If using TRANSACT, the only entries made in DICTIONARY should be those of the VPLUS file and the forms within it. No attempt should be made at this stage to define data bases, elements etc.

Structured Prototype: work study

One of the most valid reasons for doing a simulation or prototype is to get feedback from users on natural methods of working. Activity Analysis enables business activities to be successively decomposed but is not very good at describing activities enforced by logic checks. For example, if a user is entering invoices and the supplier is not yet on file, it would be nice to be able to add the supplier at this point and resume, rather than backing out to MENU-SCREEN, down to ADD-SUPPLIER, back up to MENU-SCREEN and then down again to INPUT-INVOICE.

Many designers have little understanding of user working methods. It is quite common to see all file-adds in one structure block, all file-updates in another, all file-deletes in a third block. A few minutes thought would lead designers to recognise what an unfriendly system they have created.

When you are investigating the system, ask to spend some time in the user department and study the work flow. Get the designer to watch the sequence of events in the current method of working and relate this to transactions in the new system. You will find a transaction breaks down into manual tasks (eg file the invoice) as well as computer tasks. Fit long response program actions into the work flow at periods where the user is doing manual tasks.

A useful method of anticipating system response (also useful in other areas) is Queueing Theory. Using queueing theory and work study together you can quite easily calculate how many input people are needed for a given number of transactions, given peaks etc.

Much is written about the ergonomics of hardware design - is the terminal keyboard at the right height etc. Little thought is given to software ergonomics, yet poor software design is probably the cause of more user strain than poor hardware design.

Try to be consistent without being unnecessarily rigid. Use standard approaches for function key usage, 'HELP' screens etc. One neat trick with help screens is to drive them from the procedure documentation within the data dictionary.

Some designers seem to think users like working in batches - this is patently untrue. Users cannot see why computer people force them to add things up and log in batch control books! Unfortunately there are few enlightened software designers about; even HP's new accounting software (HPFA) fails to make online updating across subsystems and uses batch runs instead.

Is this the way manual double entry bookkeeping works? Have you really

Structured Prototyping: second level

Language considerations
Having gone through one level of iteration with the initial prototype it is now time to expand to the next level. This is probably the last chance you have to decide which language to use for the system. Other than performance and speed of programming, an important decision criteria is segment/subprogram switching speed. Subprogram calls in TRANSACT are terribly slow - the only workaround is to write one large monolithic program to handle the whole system.

Terminal handler
I have assumed up to now that you will be using VPLUS. Certainly prototyping is easier to initiate using block mode, but many system builders including myself prefer character mode with formatted screens. Users also tend to prefer character mode but unfortunately VPLUS has become the standard screen handler on HP3000.

File handling
The second level prototype should still not include data base handling, however you may chose to incorporate a suitable skeleton routine which at this stage returns dummy information relevant to the particular screen. When the system goes through third level prototype, this routine would be modified to use the global IMAGE procedure rather than returning dummy data.

Second level tasks

Complete the screen/function hierarchy
Examine methods of streaming and MPE interface (eg QUERY)
Complete as much as possible of the screen detail
Determine procedure sharing amongst functions (eg IMAGE/KSAM calls)
Group functions for segmentation/subprogram structure

The second level prototype should then be demonstrated to the user and preferably left for them to play with. This is probably the last time you will get major function structure changes (ie knock out this facility, combine these two into one, add this function).

Only at this stage need the user commit to a particular system design. Once this has been accepted by the user we can move on to the third level of prototyping.

In the meantime we have by default already started our Structured Testing activities. By using the hardness/skeleton which will form part of the

Structured Prototyping: third level

We now need to convert our prototype to a working system. During the two stages of user iteration the Data Model and other documentation should be kept up to date. We should now be able to fully define attributes (fields) in terms of their type, storage length, length on VPLUS screen; the Element definitions in the data dictionary are now updated. The definition of the finalised VPLUS forms are input, cross referencing the data elements used on the forms.

We can now finalise the Entity/Attribute matrices. These are used in the design the data base but our next activity is to examine the prototype functions in light of the data model and document access path requirements. Access Path Analysis will be covered in more detail in the next paper, but briefly the objective is to document entity navigation, search keys used, frequency of access, speed of response required, numbers of entity occurences involved in order to make subjective decisions regarding IMAGE paths.

Having deciphered this information we can map our refined data model to IMAGE (also covered in the next paper) and create the data base. Finally we can start the implementation phase proper and our main task is now to remove the dummy stubs and replace with DATA BASE handling calls, preferably through a centralised routine (DBACCESS). There are other tasks, for example, defining the locking strategy

The dictionary should be used at this stage to cross reference Procedures and Elements for future 'where used?' enquiries. Hewlett-Packard are rumoured to be developing a routine to build copylibs from dictionary. As we couldn't wait we wrote our own in-house system. The advantage of generating copylibs from dictionary is the control you can now exercise over programmers and the structures they access. It is impossible to enforce the Procedure/Element cross references in dictionary without copylib generation.

The development emphasis should be on those functions which will allow us to start the file creation, again using top down testing as we proceed. When modules pass final tests they can be released to the user on a controlled basis. In this manner the actual handover may stretch for several months, but from the user's viewpoint will probably fit in better with his current workload.

Conclusion

The methods I have described owe less allegiance to prototyping theory (as used in Decision Support Systems) and more to the application of Structured Analysis and Design. Nevertheless, my main contention is that uncontrolled prototyping is unnecessary when developing a Routine DP system due to the easily defined nature of the system.

The main use of prototyping is to ensure the system is egonomically correct and models efficient work patterns. Cynically, prototyping is also an excellent method for obtaining user involvement.

Bibliography, references, suggested reading (publication date order)

1. "Adaptive design for Decision Support Systems"
Peter Keen: ACM SIGOA Vol 1 No 4/5 Sep/Nov 1980

2. "Data Analysis - The Answer to Successful Implementation of IMAGE"
Richard Irwin: HPIUG Berlin proceedings October 1981

3. "This is IT: A Manager's Guide to Information Technology"
John Eaton & Jeremy Smithers: Philip Allen Publishers Ltd 1982

4. "Management: Key to Successful Systems Implementation"
Gary Langenwalter: HPIUG San Antonio proceedings March 1982

5. "User Friendly Software Development"
Ivan Rosenberg: HPIUG Montreal proceedings April 1983

6. "Software Prototyping: Today's Approach to Information Systems Design"
Orly Larsen: HPIUG Montreal proceedings April 1983

7. "Prototyping Interactive Information Systems"
Mason/Carey: ACM Communications Vol 26 No 5 May 1983

Database Consultants europe b.v.
Keizersgracht 557, 1017 DR Amsterdam. Telefoon +31 20 - 22 42 43

IMAGE DESIGN: Logical Data Base Mapping

A presentation to the
HP3000 International Users Group
European 1983 Conference
Edinburgh October 3rd - 6th

Timothy R. Cullis
and Richard Irwin

Database Consultants Europe
69, Grasmere Gardens
Harrow Weald, Middx.
HA3 7PS England
Tel: (01) 863 2428

Abstract

This is the final paper in the IMAGE design trilogy. Having covered various structured analysis methods in the two previous papers, we now look at ways to map our conceptual thoughts to a firm IMAGE data base design. As a case study for the data base design, we have used retrospective ideas for a system to handle HPIUG conferences.

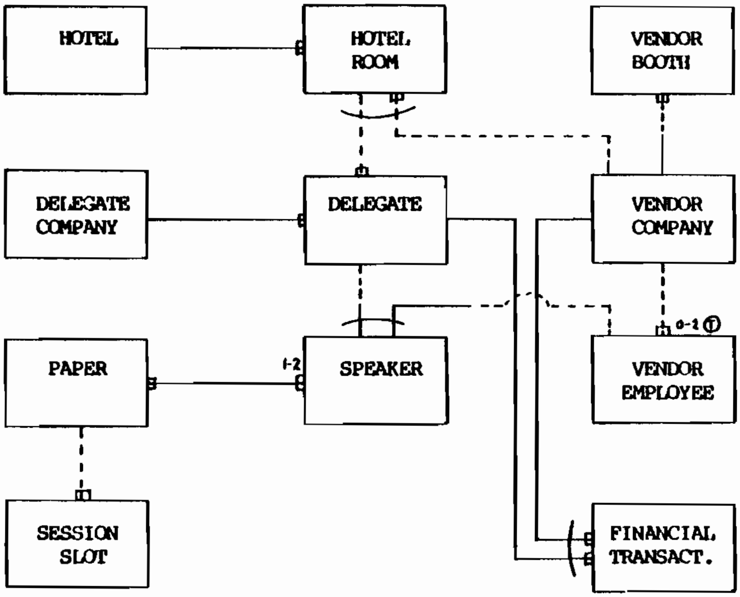
Historically, there is little continuity of experience from one conference to another either from the host committee, the IUG board or even from IUG employees. Twice a year in North America and Europe, conference host committees reinvent the wheel. By the time the host committee appreciate better ways of doing things, the next conference has already passed the relevant planning stage.

We decided (in the Edinburgh host committee) that we would document our experiences in organising this conference; this paper is one step towards the documentation.

In practise, our conference organisation is being run on a mixture of HP3000 accounting software, IBM S/34 hotel bookings system and HP100 Condor based speaker/session control system! Looking at the first pass Data Model overleaf we recognise the data area (and therefore our administration) is more complex than necessary.

First pass Data Model

The quickest way to get a understanding of a situation is to construct a Data Model. This is the first pass and took about 30 minutes of thinking and sketching. It already embodies some recognition of activities - the SPEAKER entity has been separated from DELEGATE and VENDOR EMPLOYEE as at the time we need SPEAKER information we do not have the other data available.


HOTEL
HOTEL ROOM
VENDOR BOOTH
DELEGATE COMPANY
DELEGATE
VENDOR COMPANY
PAPER
1-2
SPEAKER
0-1 ⓣ
VENDOR EMPLOYEE
SESSION SLOT
FINANCIAL TRANSACT.

As we get nearer the conference, our ideas will be refined. Initially we thought the relationship between HOTEL ROOM and DELEGATE would be 1:1 however due to the success of bookings this is no longer the case and some delegates are doubling up!

Our outline ideas at the time the papers go to press will undoubtedly have altered by October. The full text of this session will therefore be available as laser printer output and will include:

Conference Data Model and Data Flow Diagrams
Activity Definitions and Access Path requirements
Data base mapping rules for IMAGE/3000
Finalised data base design including physical design considerations

A KSAM Handler

Bruno H. Freudenthaler
HP Vienna, SEO Department

A. Reasons for the KSAM Handler

The project to develop a KSAM Handler was initiated by a software house which implemented a hardware-independent application on the HP/3000 containing the following features:

1. The "data base" consists of approximately eighty (80) KSAM files altogether.
2. The users run a large set of program in session mode using process handling (UMAIN+USONM).
3. The users frequently create different processes and exit again.
4. User logging is invoked for every file in every user process.

Running the workload described above on about 30 terminals on an HP/3000 model 44 with 3 megabyte of memory, 1 drive 7925 and 2 drives 7933 revealed the following problems:

1. Several MPE Operating System table limits were reached or even exceeded (user log id's, DST etc.).
2. The startup time of a single process turned out to be very long (at least 50 seconds, containing 6 to 8 seconds of actual creation/activation time) due to the fact that a large number of KSAM files are opened immediately after creation/activation of the process. The "logical" open of a KSAM file consists of at least three (3) physical file open's which again keeps the system disc busy.
3. The locking strategy has been implemented by using exclusive file opens creating additional disc I/O's.

For that reason a "KSAM Handler Subsystem" has been developed by the System Engineering Department of HP Vienna.

B. Overview of the KSAM Handler

The KSAM Handler subsystem provides an easy programmatic access to KSAM files.

First of all, it eliminates a part of the overhead generated by using conventional KSAM intrinsics, especially when opening or closing KSAM files.

Secondly, it not only allows conditional file locking, but also conditional record locking, a feature not normally supported by the KSAM subsystem.

Also, it optionally performs logging concerning all transactions that somehow modify the contents of the data file.

A configuration utility (KSAMCONF) allows the operator to include KSAM files for being handled by the KSAM handler or to change subsystem parameters.

An initialization routine (KSAMLORD) starts one process per configured KSAM file and allows the operator to obtain information about the active handlers and users respectively.

A cleanup routine (KSAMKILL) performs the removal of users in case of unexpected user process aborts.

A handler process is started per configured KSAM file and performs all file access functions. User commands and data are exchanged via message files.

The user communication procedures allow the user program to perform the communication with the KSAM handler(s).

Additionally, a second layer of user communication procedures may be used by the application programmer(s) which checks if the KSAM handlers are activated. If not, the procedures open the KSAM file(s) directly.

Using this layered implementation philosophy plus a small set of operating system dependent procedures in "privileged mode" all application programs remain object-code compatible and moreof the handler is totally transparent which has been a major design goal.

Further on, we need the following files for the subsystem:

1. The MASTER-file contains all configuration parameters (e.g., names of "data bases", number and names of KSAM files, table information for active users or files, log-id's etc.).

2. A set of message files is used for transmitting the requests to the different handler processes.

3. A similar set of message files is used for transmitting

the answers to the different users.

4. In case of application process aborts or similar unforeseen events a dedicated message file is used for transmitting the process-specific socalled "user numbers" (serving the purpose of identification) to the cleanup routine; this cleanup routine performs the task of setting the user numbers free.

C. How it works

The user routines basically fulfil the following tasks:

1. Open the MASTER-file and obtain a user number.

First of all the user job control words dedicated to the two possible process levels of a job or session are checked if they are established. If not, they are created and set to zero; if yes, they are tested if they are set to zero. If they contain any other value but zero it has to be a user number that could not be set free before. In that case, a message is sent to the KILLUSER message file which makes the KSAMKILL process resume execution and perform the abort of a user number by sending a "remove user x" message to each active handler, locking the MASTER-file, clearing the "active user bit", unlocking the MASTER-file and issuing the next read request against the KILLUSER message file which sets the KSAMKILL into a "blocked for I/O" state.

The MASTER-file is opened with the GMULTI option. In order to obtain a unique user number the MASTER-file is locked and the "active user bit map" is scanned for a free user number and the corresponding bit is set "active" hereafter followed by the unlocking of the MASTER-file.

After that, the user specific message file with the actual name "MESSxxxx.KSAMUSER.KSAMUTIL" is opened for input. The string "xxxx" is replaced by a four digit numeric string containing the user number obtained just before and padded by leading zeroes. Also, the user number is saved in the specific user job control word.

2. Establish the "physical" communication link to a certain handler process.

The KSAM handler directory contained in the MASTER-file is scanned for the name of the KSAM file which is to be opened. If the file name is not found, the user procedures have to provide the necessary KSAM calls in order to perform the file I/O. If the name is found in the directory, the corresponding message file of the handler is opened and the further I/O requests will be routed to the handlers via the message file. It seems worthwile noticing that the user may use any MPE file equations; all of them are resolved by our procedures. This means that any formal file name is traced back to the actual file name which is the one contained in the KSAM handler directory.

3. Establish the "logical" communication link to a certain handler process.

The user procedure sends a record to the KSAM handler message file containing the desired open option (shared or exclusive) and afterwards waits for the message sent by the KSAM handler to the user specific message file which may be either successful or unsuccessful.

4. Perform the communication.

In analogy to the "open" transaction all the other requests are exchanged between the user and the KSAM handler(s). The list of available commands is contained in the appendix.

Every KSAM handler maintains user specific information about the record numbers and chains he has accessed in the last call in the same way as the KSAM intrinsics do it. Thus the user process does not recognize the difference whether he obtained the data or return code from the MPE file system directly or from a handler.

If the users want to perform logging all transactions of the KSAM handlers involving add, update or delete are logged. For that purpose the operator only has to specify a log identifier in the MASTER-file using the utility KSAMCONF. Also, the "timer" and "trace" option may be invoked per data base. The timer option prints a usage statistics of all implemented requests, namely the number of calls, sum of CPU-time and sum of elapsed time. The trace option shows the type of request a KSAM handler has received from a particular user, but not the data. This option is especially useful for establishing a user interface or debugging purposes. The output of the trace option is sent to the operator console.

5. Close the "logical" communication link.

The transaction of closing the "logical" communication link consists of the same sequence of operations as the opening handshake; this feature shows one of the advantages over the conventional way of opening a KSAM file by avoiding the overhead of accessing the system directory etc.

6. Close the "physical" communication link.

The closing of the "physical" communication link shall only be performed once just before exiting the process or if the communication will not be needed again during the execution of the process in order to omit unnecessary overhead.

7. Return the user number and close the MASTER-file.

The procedure performing this task should be called every time before exiting a process. Its main task is to return the user number which is performed by locking the MASTER-file, clearing the specific "active user bit" and unlocking the MASTER-file. Hereafter the specific user job control word is reset to zero and the user input message file is closed.

If the process aborts unexpectedly, the user number of the process cannot be returned properly. Now, if the next process is started immediately without leaving the job/session, the starting routine takes care of that, but if the job/session is terminated this may create problems if it happens too often. For that purpose a special program may be executed which checks the user job control words and performs the same steps to clear "dead" user numbers as described under paragraph 2.

D. Preliminary Test Results

The usage of the KSAM handler showed the following preliminary test results:

1. A lot of table entries are saved (only 1 user log-id per handler, 1 KSAM file extra data segment per handler etc.).
2. The startup time is reduced by approximately 30 percent.
3. Situations involving the locking strategy improved by 40 percent and more.

It must be mentioned that up to now the emphasis has been put on debugging the software and the comparison with the conventional KSAM intrinsics has been neglected, but the comparisons will be performed in the near future.

List of commands for the KSAM-Handler:

CL ..... close (logically)
DE ..... delete keyed
DL ..... delete locked record
ER ..... erase all data within the data file
IF ..... info about open file
IL ..... info about locks
IU ..... info about user
KY ..... keyed read
LC ..... lock chained
LF ..... lock file
LK ..... lock keyed
LP ..... lock partial key
LS ..... lock serial
OP ..... open shared (logically)
OX ..... open exclusively (logically)
PK ..... partial keyed read
RC ..... read chained
RL ..... remove locks
RS ..... read serially
RU ..... remove user
RW ..... rewind serial pointer
SD ..... shutdown KSAM
UK ..... keyed update
UL ..... update locked record
UN ..... unlock file or record
US ..... update serially
WS ..... write serially
WT ..... write keyed

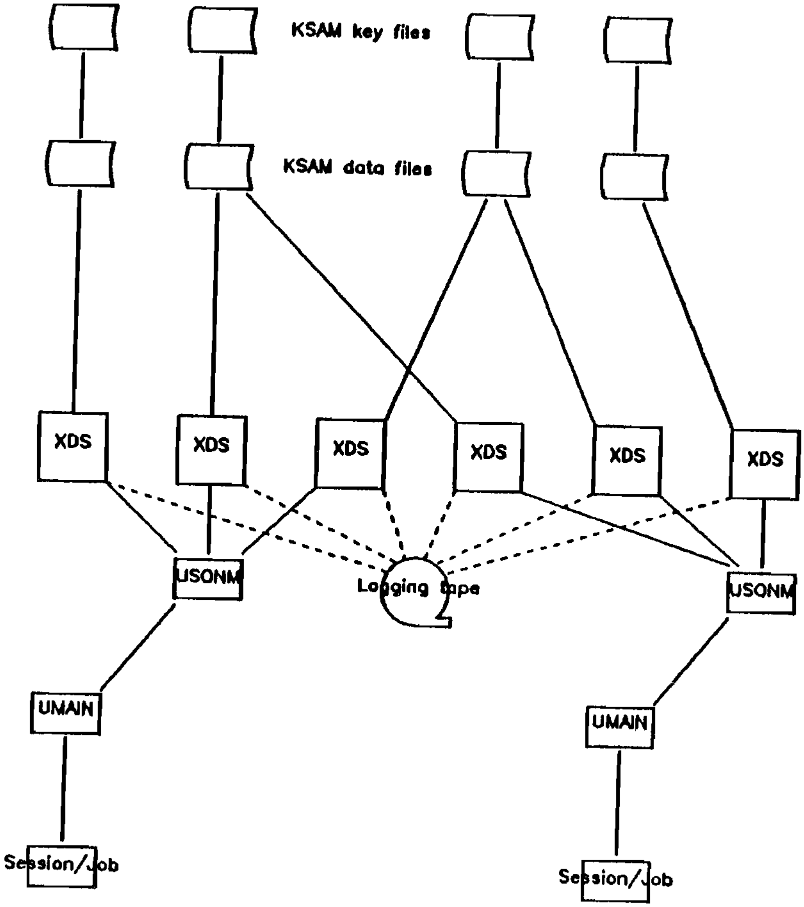

KSAM key files
KSAM data files
XDS
XDS
XDS
XDS
XDS
XDS
USONM
Logging tape
USONM
UMAIN
UMAIN
Session/Job
Session/Job

Fig. 1: Conventional KSAM file opens

KSAM Handler

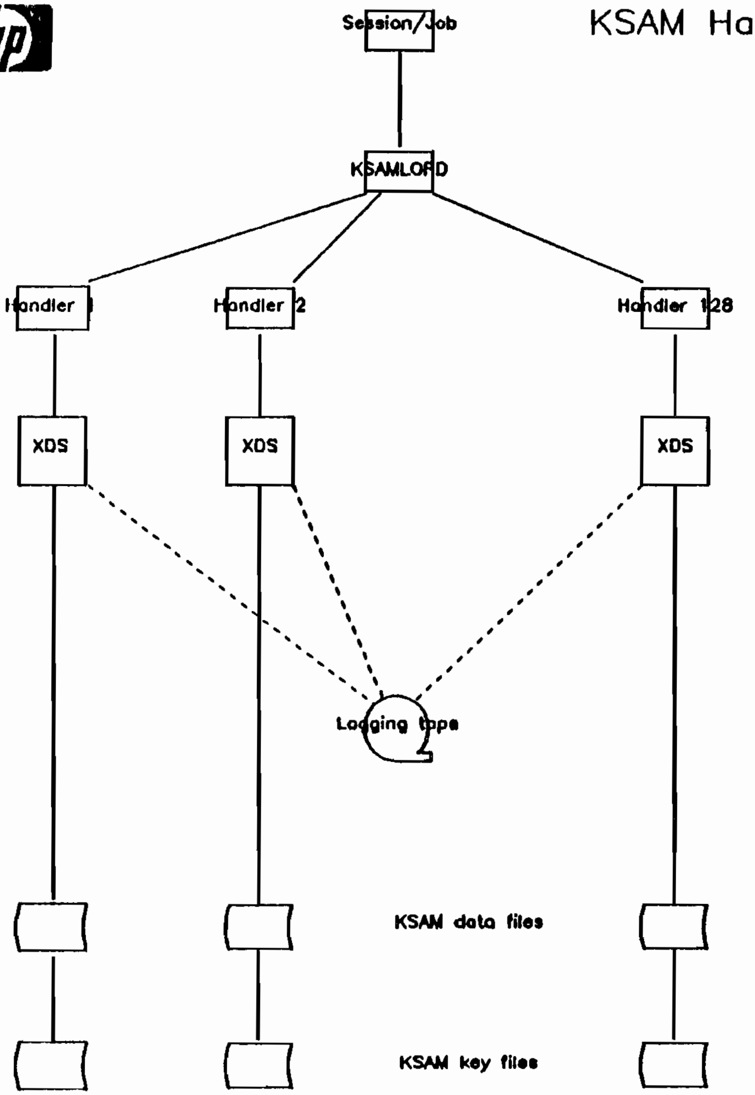

Session/Job
KSAMLORD
Handler 1
Handler 2
Handler 128
XDS
XDS
XDS
Logging tape
KSAM data files
KSAM key files

Fig. 2: KSAM files opened by handlers

KSAM Handler

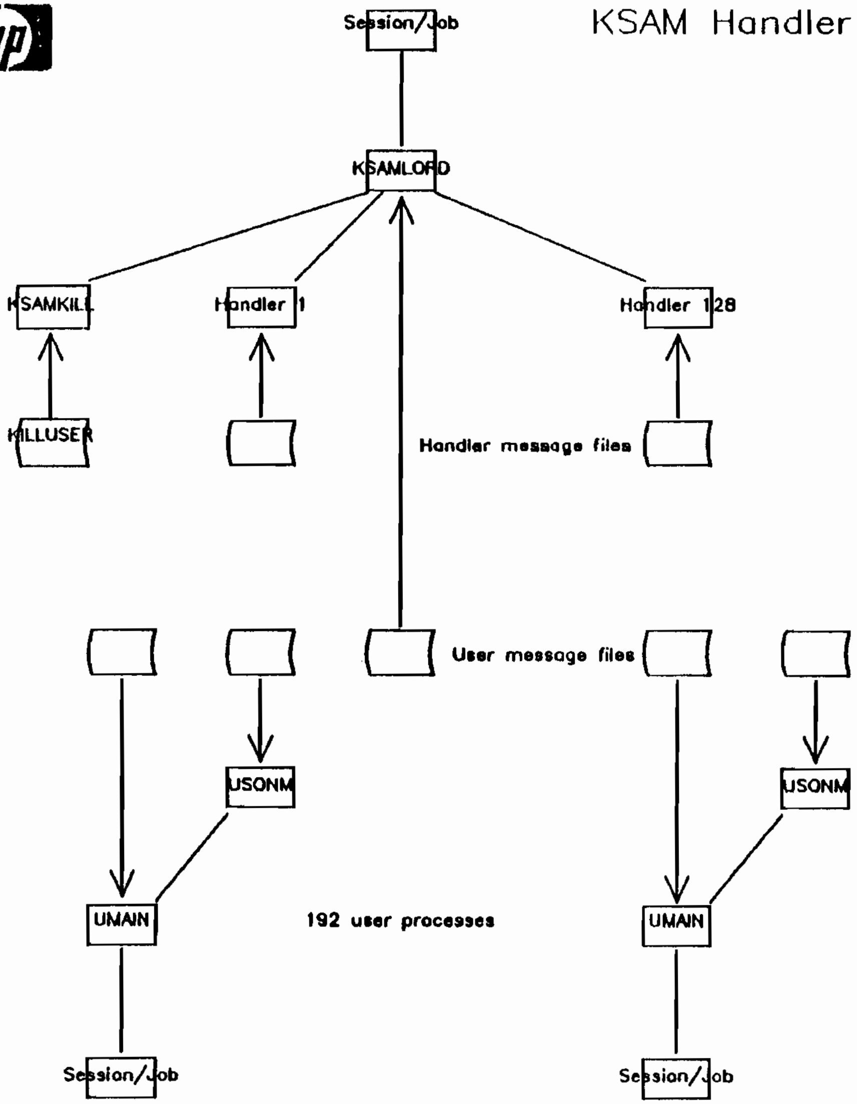

Session/Job
KSAMLORD
KSAMKILL
Handler 1
Handler 128
KILLUSER
Handler message files
User message files
USONM
USONM
UMAIN
192 user processes
UMAIN
Session/Job
Session/Job

Fig. 3: Overview of process structures and message files

TRIPLE CONVERGENCE; SCOTTISH LIBRARY INFORMATION SYSTEMS ON THE HP 3000

Bruce Royan, SCOLCAP, National Library of Scotland, UK

Abstract: Part of the challenge of the 80's is the convergence of Telecommunications, Data Processing and Information Science. The Scottish Library Network exploits this triple convergence to handle online the cataloguing, acquisitions and information retrieval needs of some 25 major libraries in Scotland. MTS/3000, RJE 3000 and x25 are utilized to run a Distributed Database System between an Edinburgh-based 2 Gigabyte HP3000 series 44 and, the British Library's large IBM machine. The scope for sharing the intellectual work of book cataloguing and classification is enormous, and this system provides both the controls and the flexibility of presentation to allow this gain in productivity to take place in institutions of differing size, constitution and outlook. Acceptance of this new technology has been eased by careful design of the man-machine interface, developing screens and character sets customized for library use. The offerings of HP provide a natural match with the automation needs of the book world and it is fitting that this system should be running in Scotland in time for IUG Edinburgh 1983

THE CONVERGENCE

More and more of our telephone calls are being switched by machines we would once have called computers. Meanwhile our computers are talking to each other over telephone lines. On the theoretical front, library scientists started to use concepts from telephony at about the same time as computer scientists began taking seriously work done by librarians on the organization of knowledge. This triple convergence of Telecommunications, Data Processing and Information Science has come not a moment too soon for a library world hard pressed to maintain bibliographic control over what is popularly known as the 'information explosion'. It is now a commonplace to point out that, for example, more scientific and technical literature has been published in the last 10 years than in the previous 10,000. What may not be so well known is the rapid growth in the number of bibliographic computer systems being developed to cope with it; during the nine months to April 1983, the number of database systems commercially available for online access worldwide increased by 30%. One system of particular relevance to a Hewlett Packard user conference in Scotland, is the subject of this paper.

A LIBRARY NETWORK FOR SCOTLAND

The Scottish Library Network has been set up to serve initially some 25 major libraries throughout Scotland and across the border in the North East of England. While 10 of the dedicated terminals are point-to-point, another 40 work on multipoint lines under MTS/3000. The problem of running so many terminals at one time with an acceptable response time has been resolved by using the package TPE (Transaction Processing Environment, supplied by Riva, of Bolton, England) which controls all terminals under a single MPE session. To allow occasional enquiry access to a further 600 terminals spread across the research establishments of the ERCC (Edinburgh Regional Computer Centre Network), we are developing an x25 gateway into the British Telecomm PSS (Packet Switched Service) Network. This part of the system has been subject to a range of delays, first from British Telecomm, then from HP, and now from our software house, but is essential in extending our coverage to places that could never afford a dedicated terminal.

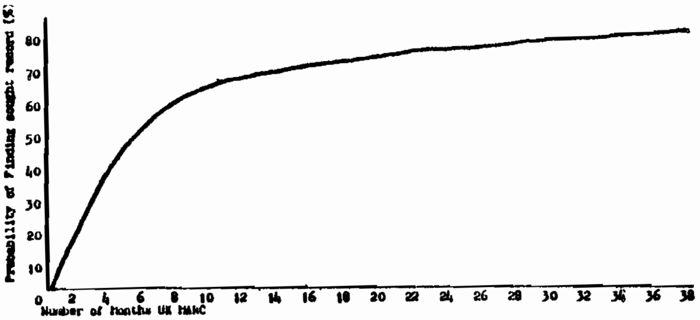
A DISTRIBUTED DATABASE

The processes of acquisitions, cataloguing and information retrieval that our system supports are based on a bibliographic record - something quite unlike the sort of records encountered in more 'normal' dataprocessing. Modelling as it does a real world, where a book may have many authors or none at all, and a title may contain five characters or 500, this record consists of variable length, variable occurrence fields. The fields themselves may be 'local' fields related only to a particular library's copy of a book (whereabouts it has been placed on the shelves, say, or whether it has been rebound), or 'general' fields relating to all copies (author, title). This structure is mapped onto the IMAGE database system as follows. Each logical record consists of a master 'Bib' record containing fixed (often truncated) versions of fields needed for key access, now or in the future, with a number of other records attached to it. There are overflow (REST-OF-BIB) records completing the variable length/variable occurrence general data available; there are local records for each library holding; and there are records relating to particular orders for the book.

The system went live with a database of well over a million holdings of 750,000 titles. This already takes a substantial portion of our 2 Gigabyte disc capacity, and our libraries continue to catalogue some 180,000 new titles a year. Added to this, our members required access to the vast (currently 4 million) and growing files of records in the MARC (Machine Readable Catalog) format created by the Library of Congress and the British Library. Fortunately, the British Library has mounted these records on a series of databases on its IBM-based British Library Automated Information Service (BLAISE). What was necessary was for our system to provide users with the record required, irrespective of where that record might be stored. The system developed (and dubbed SCOLCAP) exhibits the three classic features of a Distributed Database System.

Partitioning can be defined as "the separation of the conceptual database into increments that reside in two or more locations". We have already seen that BLAISE is considered as a source of potential requirements while SCOLCAP stores locally only the actual holdings of our libraries. The user does not need to know where a particular record is stored; this is ascertained by the system when the record has been requested.

Replication can be defined as "maintaining copies of all or part of the conceptual database, to reduce telecommunications line traffic and to increase availability and resilience". Studies of our libraries' intake have shown that a relatively small portion of the most recent British records on BLAISE, will satisfy a relatively large portion of the demand for new catalogue records:



As new records are added to BLAISE therefore, we also add them to SCOLCAP, dropping them if they have not been used within the first few months when demand is heaviest.

Fragmentation is "the distribution of the data elements of a logical record across more than one location". After a book has been ordered and catalogued, the 'volatility' of its record begins to decay. Put another way; the records most likely to be required for consultation or amendment are those that have recently been created or amended, and once through the initial active stage are unlikely ever to be touched again. When this relatively stable state has been reached therefore, the system automatically checks that the record is available on BLAISE. If it is, considerable disc space is saved by 'discarding' the general data from the SCOLCAP database (the local data continues to be held locally on SCOLCAP). Although the user continues to think of one catalogue record, it is infact distributed between two physical systems 500 miles apart.

The computer/computer communication that facilitates this distribution is effected by HPs standard 2780 emulator. This is far from ideal in an application requiring constant interaction, but was the only protocol on which all parties could reach agreement. Each line in the inter-machine conversation is treated as a file, and to prevent time-outs the SCOLCAP HP 'talks' to the BLAISE IBM machine all day long, sending regular 'do nothing' messages whenever there are no outstanding searches to be done.

All this is best illustrated by an example; say a user wants to order two books. He asks for the first by its International Standard Book Number (or Author, Title, Key words, etc). The system searches the SCOLCAP database, and if the book is not found, asks BLAISE to search its file of current British records for it, continuing to switch between files in this way until a hit is found. If this process takes longer than 6 seconds, the user is kept informed of the progress of the search; otherwise the record is displayed on the vdu without any further formality. The second search may find a fragmented record as described above. In this case the user is asked whether he wants to see what is available locally (local data plus short author/title details etc) or the full record. If the full record is required, the general data is retrieved from BLAISE and blended with the SCOLCAP-held data to create a virtual record which is presented to the user.

SHARING THE INFORMATION

This ability to present to each user his own view of the data is vital to a system such as this, the purpose of which is to share the intellectual effort of book description between many different institutions. The need for this was identified over 100 years ago:

> "When I was a Librarian myself, I always wondered at the extraordinary waste of power in cataloguing new books. While I was writing my slip according to the rules followed in most libraries, I felt that there were probably 100 people doing exactly the same work." (Prof. Max-Muller)

Nowadays the number of libraries worldwide likely to be cataloguing the same book at about the same time could be counted in thousands rather than hundreds, and the Anglo-American Cataloguing Rules are painstakingly observed by so many of them, that there are tremendous savings to be gained by allowing each catalogue entry to be made once and shared by all.

SCOLCAP's libraries come in all shapes and sizes. There are Colleges of Technology like Napier (once the home of the inventor of the slide rule), public libraries orientated towards light home reading, great and ancient university and research libraries and small specialist institutions like the Royal Observatory (once the collection of the computer pioneer Babbage). Each has its own policies towards the level and quality of description it needs from, and is prepared to contribute to, its catalogues.

The system therefore had to control who is allowed to enter or amend which field on each record to maintain its quality and integrity. This is done by means of a range of priority codes applied both to the records and individual users. This strict control is tempered by great flexibility in the presentation of the data. A table (called a Profile) is set up for each user specifying the fields that should (and should not) be displayed, prompts that should be given during data entry, validation applied, and so on, so that from the users point of view the records have been designed specifically for each libraries use..

GAINING ACCEPTANCE

Such tailoring is part of a range of design features that were necessary to ensure that the twin revolutions of automation and cooperation would be accepted by a profession perhaps a little conservative in outlook.

Considerable care was taken with the choice of the vdu, and then with the design of the screens that were to appear upon it. Long before the 3000 had been installed, a simulation of the vdu conversations was set up on a 2645A with cartridges, and was taken around all the libraries for comment, the feed back from which was immensely useful in finalizing the screen layouts. The only drawback, was that many users became convinced that the system was at that stage already up and working, and could not understand the 'delay' in its implementation!

Similar attention was paid to the character set. Libraries deal with a range of material, including scientific and technical texts and non european literature, which requires characters beyond the range of standard ASCII. After long consultation we developed a special character set for libraries. It contains all of USASCII plus Greek upper and lower case and a range of extra national (Scandinavian, Polish, Turkish) characters, plus diacriticals (accents). With this set it is possible to catalogue all material either directly or according to internationally accepted rules of transliteration (eg Cyrillic). The 264xx's matrix of 7 x 13 with half shift proved totally adequate for defining the characters required. The only difficulty encountered was with the superimposition of diacriticals. In normal printing, a diacritical superimposes the character it modifies (eg è, ê). Unfortunately there seems to be no true 'overstrike' facility on the 264xx; one cannot simply 'or' the matrices together. The solution was to define a third character set (in addition to the 'standard' and 'special' sets) containing all known combinations of character and diacritical, and modify the terminal firmware to display from this set whenever a character has been preceded by a legal diacritical.

Our experience with the character set is an example of something more general. The requirements of an online distributed cooperative bibliographic automation system have proved to be specialized, even outlandish (at one agency's office I recently saw a poster: GOD IS ALIVE AND WELL AND WORKING ON SOMETHING LESS COMPLICATED THAN SCOLCAP), but they have always, so far, been matched by some feature of HP hardware and software. In working on this project I have found out about a wide range of book trade systems that are using the 3000 as their workhorse. Perhaps the time is right for a Special Interest Group on bibliographic systems within HPIUG. If I can think of a suitable acronym, perhaps I'll set one up .....

A TREE GROWS IN IMAGE
Data Structures in the IMAGE Database

Theodore Dillenkofer Jr.
Software Development Manager
Austin Information Systems, U.S.A.

Introduction

Information aquisition, maintenance and access has become on of the foremost uses of computing machinery & software. Tools to organise information have been developed and utilized with varying degrees of effeciency. Implicit in the development of these tools and the systems that employ them has been the concept of data structures; the logical organization of data into maintainable, accessable formats for use in computerized information systems. This paper describes and explores in depth the hierarchal data structure commonly referred to as a tree and how to implement a hierarchal model using the capabilities of the IMAGE database.

Nomenclature

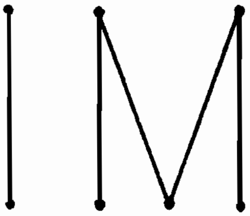


Figure 1

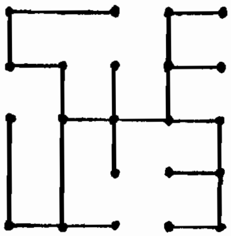


Figure 2

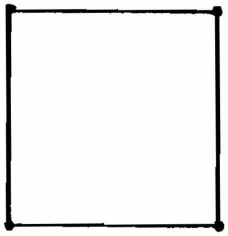


Figure 3

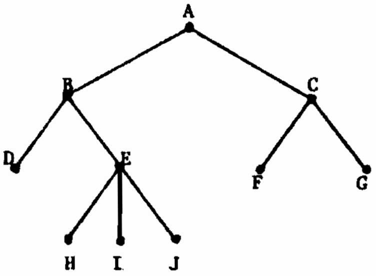


Figure 4

Terminology and morphology of trees can best be communicated through the use of graphs. A graph is a set of points (NODES) defined by the termination vertices of descreet line segements (EDGES or PATHS). Figure 1 is a graph with seven nodes and five paths. A special case is presented in Figure 2, the connected graph. A connected graph is any graph in which it is possible to contruct a path between any two vertices by using existing paths and nodes. Figure 3 is a subset of connected graphs known as a circular or cyclical graph. In this case, a path can be contructed using three or more nodes to return to the starting node. This structure must be avoided when using the technique in this paper for reasons which are explained in the section on traversal algorithms.

Figure 4 represents a tree structure. A tree can be defined as a connected graph with no circular structures. In this paper we will deal with oriented trees. An oriented tree is a tree in which one node is identified as the root node (node A in figure 4). The other nodes can be referred to in terms of levels. The root node is level zero. Nodes B and C are on level one, nodes D, E, F and G are on level two and H, I, and J are on level three. A particular node is referred to in terms relative to other nodes on the path between that node and the root. Any nodes on the path between the root node and the particular node are referred to as ancestors. The immediately preceding node is the parent. Any nodes further down the tree from a given node are called descendants. Descendants immediately following a node are called children and a group of children with the same parent is referred to as a family. A node with no descendants is called a terminal node. Referring again to Figure 4, node A is the root node with children B and C. B is the parent node of a family composed of two children, D and E, of which D is a terminal node and E is a parent in its own right. The ancestors of node I are B and E.

There is another form of tree called a binary tree, which is a subset of the general tree graph. In a binary tree, each node has one parent and at most two children. Removing node I from Figure 4 produces a binary tree. While binary trees can be used to represent the same structure as a regular tree, we shall use the regular tree for the purposes of demonstration.

<u>IMAGE model</u>

```
NAME: PARENT-MAST, MANUAL;
ENTRY: PARENT(1);
CAPACITY: n;

NAME: CHILD-DTL, DETAIL;
ENTRY: PARENT(!PARENT-MAST);
       CHILD;
CAPACITY: n;
```

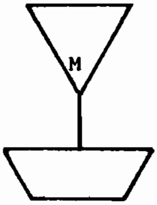


Figure 5

Implementing a tree utilizing IMAGE is a simple procedure. Two data sets are required: a manual master called PARENT-MAST and a detail set called CHILD-DTL. This is the minimum configuration ; other data sets can be added to simplify operations and expand capabilities as explained in the applications examples. Figure 5 shows the database morphology and data items in each data set.

Each family in the tree is entered as an IMAGE chain. The parent node is put in the PARENT-MAST and each child is entered into the CHILD-DTL with itsparent node as the search item. Some nodes are both parent and child (nodes B, C and E in Figure 4) and are entered both in PARENT-MAST as PARENT and in CHILD-MAST as the CHILD of its PARENT node and as the PARENT ot its own children (Figure 6).

PARENT-MAST

PARENT
A
B
C
E

CHILD-DTL

PARENT	CHILD
A	B
A	C
B	D
B	E
C	F
C	G
E	H
E	I
E	J

Figure 6

In this manner a tree of any length can be constructed, limited only by the capacity of the data sets.

Traversal algorithm

One of the most important procedures performed on a tree is a traversal. A traversal is accomplished by starting at the root and following every path to every node in the tree. In our example tree of Figure 4, the proper traversal sequence is:

A, B, D, E, H, I, J, C, F, G

This is essentially left to right, top to bottom. Accomplishing this traversal requires the use of IMAGE's internal chain pointers, accessed through the database control block as previous record number, next record number and current record number. A stack or array must be set up to hold the pointer of the current record being accessed in a mode 5 DBGET and the chain head. Figure 7 is a flowchart of the complete process.

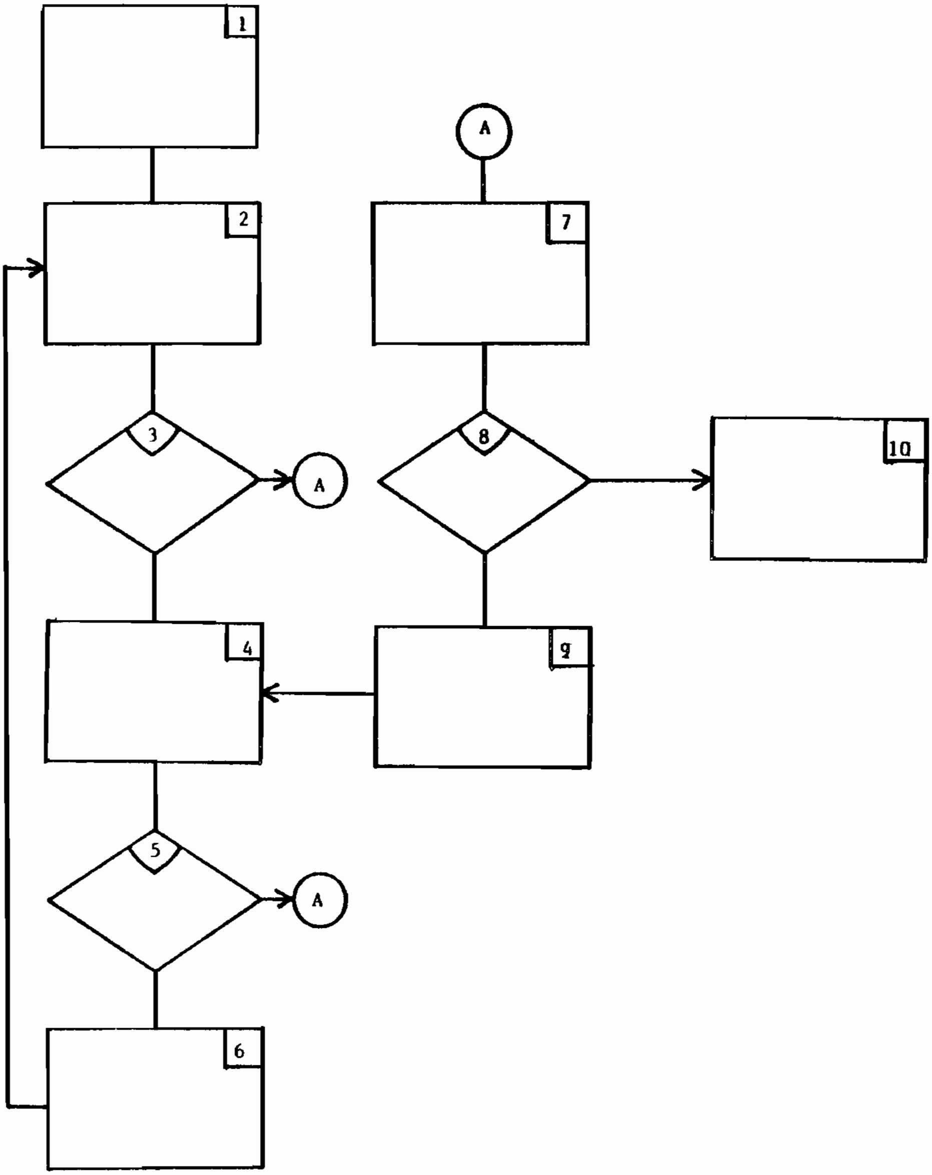

1
2
3
4
5
6
7
8
9
10
A
A
A

Figure 7

STEP 1: Initialization.
The array to hold chain heads and pointers is set up. The dimensions are (2,n), where n is the largest number of levels expected. The level counter (LEVEL) is set to one, the array is initialized to blanks and the root node is obtained and moved to ARRAY(1,LEVEL) and the variable PARENT.

STEP 2: DBFIND.
A DBFIND is performed on the CHILD-DTL using PARENT as the search item value.

STEP 3: Check for descendants.
The number of records in the chain is checked through use of the NUM-RECS variable in the DB control block. If the current node is a terminal node, the procedure jumps to a routine which reestablishes position in the preceding chain, then returns to step 4. This recovery procedure is covered in steps 7 through 10.

STEP 4: DBGET.
A chained (mode 5) DBGET is performed on the CHILD-DTL, bringing a child into the procedure.

STEP 5: Check for end of chain.
If the end of the chain has been reached (DBCW=+15) the preceding chain must be reestablished. This is accomplished using the same routine branched to in step three and returns to step 4.

STEP 6: Storing information.
ARRAY(2,LEVEL) is set to the current record number of the chain using CURR-REC-NO of the DB control block. This places the chain head and the position in the chain opposite each other in ARRAY. LEVEL is incremented by one, the CHILD from step 4 is moved to ARRAY(1,LEVEL) and variable PARENT is equated with CHILD. Process control is then shifted back to step 2.

STEP 7: Reestablishing the chain: part 1.
LEVEL is decremented to serve as a pointer into ARRAY for the previous chain head and record pointer.

STEP 8: Check for finish.
If LEVEL < one, then all nodes have been accessed and the procedure is finished. Step 10 terminates the procedure.

STEP 9: Reestablishing the chain: part 2.
Call DBFIND using ARRAY(1,LEVEL) as the search item value. Use a directed read (mode 4) DBGET using the pointer value in ARRAY(2,LEVEL) as the record number. This raises the level of access by one and resumes the previous chain at step 6 before the branch to step 2 took place. The routine then jumps to step 4, getting the next entry in the chain.

STEP 10: End of traversal.

Any operations to be performed on data should be inserted in step 6 before the branch to step 2. An example using this technique to store and retrieve parts lists and generate total part quantities follows.

Example: Product Structure System.

```
BEGIN DATA BASE PSBASE;
PASSWORDS: 10  BOSS;
ITEMS:  PART,            X16;
        COMP,            X16;
        DESC,            X20;
        QUANT,           I1 ;
        SUM,             I2 ;
        LEVEL,           X2 ;
SETS:
 NAME:  PART-MAST,MANUAL(10/10);
 ENTRY: PART(1),
        DESC,
        SUM,
        LEVEL;
 CAPACITY: 50;
<<  >>
 NAME:  COMP-MAST,AUTOMATIC(10/10);
 ENTRY: COMP(1);
 CAPACITY: 50;
<<  >>
 NAME:  COMP-DTL,DETAIL(10/10);
 ENTRY: PART(!PART-MAST(COMP)),
        COMP(COMP-MAST),
        QUANT;
 CAPACITY: 50;
```

Figure 8

```
END.
```

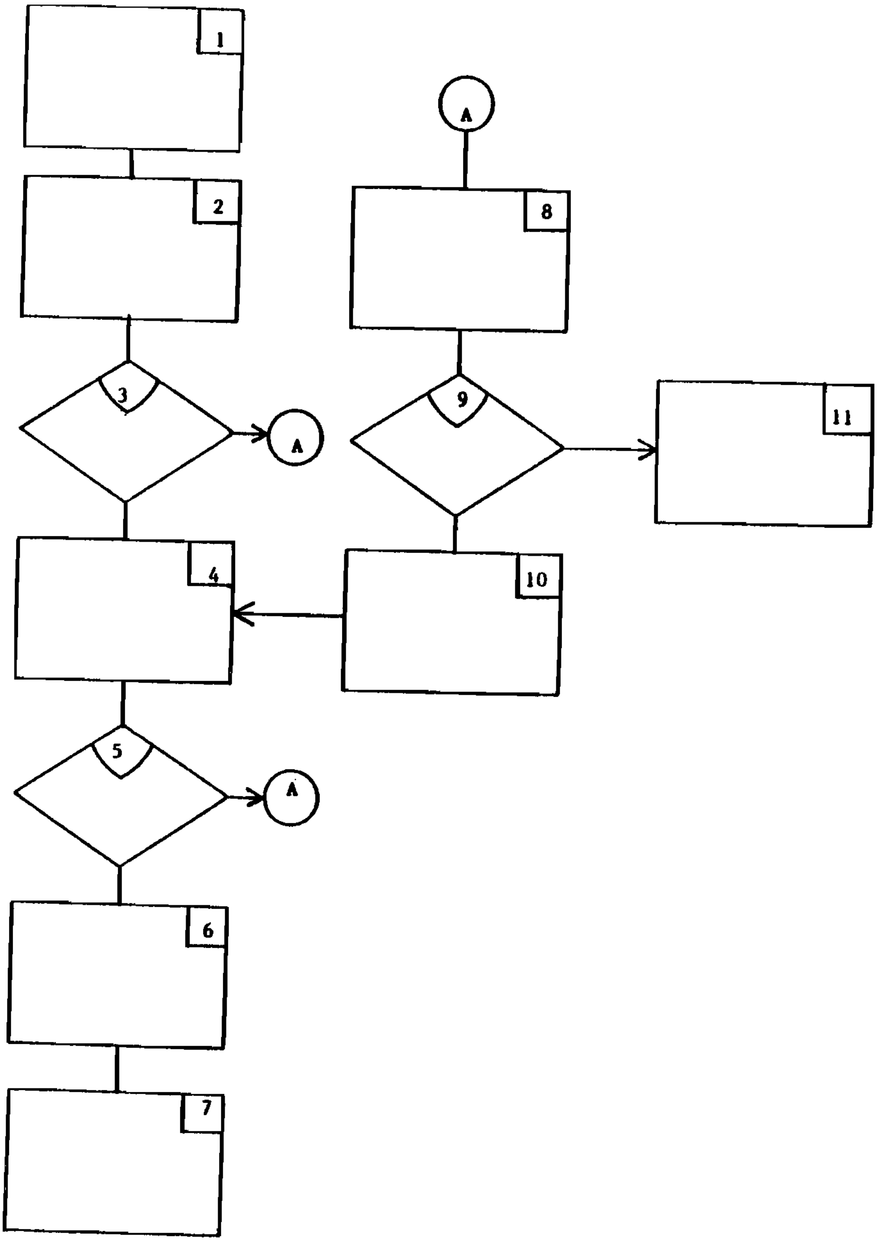

1
2
3
A
4
5
A
6
7
A
8
9
11
10

Figure 9

The first step is to create individual parts lists and then link them to the top assembly to form a tree. If QUERY is to be used, all parts must first be added to the PART-MAST and then added to the COMP-DTL, using the assembly part number as PART, the component part number as COMP and the quantity of components as QUANT. One record is added to the detail set for each component of the assembly.

When each assembly is complete, it must be linked to the next higher assembly in a similiar fashion. Add the next higher assembly number to PART-MAST, then add one record for each sub-assembly as was done for components. This process continues until all components, sub-assemblies, and assemblies are entered into the database, forming a tree with the top assembly number being the root node.

At this time, PART-MAST contains a list of all part numbers used in the product, COMP-MAST contsins all the entries in PART-MAST except the root or top assembly number and the COMP-DTL holds the structure of the tree. It is important to note that any given part can occur in any number of assemblies, reducing the amount of data entry time and disc space required to build and store the tree.

Parts lists can be generated by performing a DBFIND on COMP-DTL using any PART value as the search value for the search item PART. A chained DBGET (Mode 5) results in a list of all components used in the assembly requested.

In a like manner, where used lists can be obtained using a DBFIND on COMP-DTL with any COMP value as the search value for the search item COMP. This procedure results in a list of assemblies where a part is used. To obtain full lists of either assemblies or where used, the PART-MAST or COMP-MAST must be read serially, performing DBFIND's and chained DBGET's on the COMP-DTL using the previously mentioned techniques. The masters can also be sorted to make the resulting lists easier to user.

One of the major uses of systems of this nature is to generate gross requirements of parts required to build a product. This is the purpose of the SUM field in the PART-MAST. A tree traversal is performed and each QUANT in the COMP-DTL is multiplied by the QUANT's in the brach it belongs to and added to the SUM of that QUANT PART's PART-MAST record. Figure 9 is a flowchart of the algorithm to calculate these sums. The superstructure of the flowchart is identical with that of Figure 7, differing only in the structure of ARRAY, a few extra varibles and some IMAGE calls to DBGET and DBUPDATE the PART-MAST records.

directed read (mode 4) DBGET using the pointer value in ARRAY(2,LEVEL) as the record number. This raises the level of access by one and resumes the previous chain at step 6 before the branch to step 2 took place. Divide LEVEL-MULT by ARRAY(3,LEVEL). The routine then jumps to step 4, getting the next entry in the chain.

STEP 11: End of traversal.

STEP 1: Initialization.
The array to hold chain heads and pointers is set up. The dimensions are (3,n), where n is the largest number of levels expected. The level counter (LEVEL) is set to one, the array is initialized to blanks and the root node is obtained and moved to ARRAY(1,LEVEL) and the variable PARENT. The quantity multiplier (LEVEL-MULT) is set to one, as is ARRAY(3,LEVEL).

STEP 2: DBFIND.
A DBFIND is performed on the CHILD-DTL using PARENT as the search item value.

STEP 3: Check for descendants.
The number of records in the chain is checked through use of the NUM-RECS variable in the DB control block. If the current node is a terminal node, the procedure jumps to a routine which reestablishes position in the preceding chain, then returns to step 4. This recovery procedure is covered in steps 8 through 11.

STEP 4: DBGET.
A chained (mode 5) DBGET is performed on the CHILD-DTL, bringing a child into the procedure.

STEP 5: Check for end of chain.
If the end of the chain has been reached (DBCW=+15) the preceding chain must be reestablished. This is accomplished using the same routine branched to in step three and returns to step 4.

STEP 6: Storing information.
ARRAY(2,LEVEL) is set to the current record number of the chain using CURR-REC-NO of the DB control block. This places the chain head and the position in the chain opposite each other in ARRAY. LEVEL is incremented by one, the CHILD from step 4 is moved to ARRAY(1,LEVEL) and variable PARENT is equated with CHILD.

STEP 7: Updating the sum field.
DBGET, mode 7 on PART-MAST. Move QUANT to SUM. DBUPDATE on PART-MAST. Move QUANT to ARRAY(3,LEVEL). Multiply LEVEL-MULT by QUANT. Process control is shifted to step 2.

STEP 8: Reestablishing the chain: part 1.
LEVEL is decremented to serve as a pointer into ARRAY for the previous chain head and record pointer.

STEP 9: Check for finish.
If LEVEL < one, then all nodes have been accessed and the procedure is finished. Step 10 terminates the procedure.

STEP 10: Reestablishing the chain: part 2.
Call DBFIND using ARRAY(1,LEVEL) as the search item value. Use a

HOW TO GET A HIGH PERFORMANCE ORDER-FILE

by
Hanne Hansen, M.Sc
DOMI A/S
Denmark

Abstract

How do you organize an order-file, when you have to add 5000 new orderlines a day, and still be able to do direct access to the orderlines ? The first answer is naturally KSAM or IMAGE. We thought it to be KSAM, but after we started production on the first part of the system, we found that production on the whole system, probably would give unsatisfactory responsetimes. The second answer and the answer we stopped with, is a structured MPE-file. This paper describes how we have structered the file, from our exact knowledge of the data to be stored in the file, how we have made intrinsics for accessing the file, including generic search, and how we have made it possible to change the key values, without the expensive disc I-O you have with KSAM.

The System

The system we needed, was a order-entry system, capabel of handling 5000 orderlines a day. The entry of the orders would not present any problems, but as the orders normally passes thru the whole system on the same day they are entered, we found that direct-access to the orders, was a must.

The problems

We found, from our knowledge of the HP-3000, that using a KSAM-file for order/orderline-file, would be the right solution, but when we started production on only a part of the system, we realized that if we were to run the whole system, we would get unsatisfactory responsetimes.

Knowing this, we started looking for the bottleneck of the system, and soon found, by examine the logfiles, the KSAM-file to be the main problem. From the logfiles, we saw, that the number of accesses on the key-file was 8-20 times higher than on the data-file. As we investigated the problems further, we found that the responsetimes varied considerably, from user to user, but seemed to be stable for each user. As it turned out, the answer was quite simple. Some of the users entered one order, with all connected orderlines, at the same time. In this case the order and the orderlines would be written close to each other in the KSAM-file, whereas others entered the orders in the morning, and then added orderlines to all the orders, during the day. In this case the order and the lines are spread all over the file, thus giving longer responsetimes, when the order, later on, was processed. The elapsed time used for adding one transaction, one orderline, varied from 1.9 sec. for the first type of user to 3.0 sec. for the second type.

The solution

With this new information, and some extra demands to a new orderfile, we felt that we would be able to build a normal MPE-file, and then logically structure the file to solve the same problem as KSAM did in the original version, but with improved capabilities and reduced responsetimes. As we started working on the idea, we found that, by making two different structures in the MPE-file, we could replace both the KSAM-file, and the IMAGE databases, we used in the system.

The logical D-File

The file consists of two psysical files, a data-file and a key-file. The first records of the data-file holds a description of the logical files contained in the file, where the keys are placed in the records, and some pointer information (first entry etc.) The first records of the key-file contains information on the keys in the file, size, position etc. and links to the last sorted key and the last written key.

Our main problem with KSAM, in this environment, is the heavy access on the key-file, because we, to be sure of which record is current, has to read thru the key-file, even for 'sequential' access. To solve this problem, we specified, in the data-file definition, how many records to allocate per primary key entry. (A definition of 1 record per primary key will give you a normal 'KSAM-type' file) When the first record, with a specific key value, is written, a number of contigous records are allocated in the data-file. Any subsequent writes, with the same key value, will be written in the block allocated by the first write, and no key entry will be added in the key-file. With this structure, a whole datablock can be retrieved with only one access to the key-file.

This first version contains fixed length data records of 128 words each, and variable length keys, where, similar to the UNIX concept, just the necessary number of words (bytes) are written. A later edition will contain the same concept for the data-file.

The intrinsics

To manipulate the data we have made the following intrinsics :

- DREAD	will read the next record in the data-file in key order.
- DREADBYKEY	will read the first record in the data-file, with a specific key value.
- DREADDIR	will read a record in the data-file, specified by recordnumber.

- DUPDATE	will update current record in the data-file. Any keys, referring to only one record, can be changed.
- DWRITE	will write a new record in the data-file.
- DREMOVE	will remove current record from the data-file and from the key-file.
- DREORG	will remove all deleted records in the data-file and sort all key entries into a new file.

Data consistency

The sequence of operation, is to update the key-file before the data-file. For DUPDATE, if a key value has been changed, the sequence is : A new key entry is written, the data record with the new key value is updated, the old key entry is deleted. This sequence, and the fact that any mismatch, between the key value in the key-file, and the key value in the data-file, is considered as 'record not found',(The entry is logged and removed) releases you from the time consuming recovery from system failures, as KSAM-files requires. The file will be recovered automatically, and you might only loose what was current record.

Reorganization

As new key entries are added to an overflow area of the key-file, it means that you have to reorganize the file, when the overflow area becomes too big. By doing this, rather than, like KSAM, keep the file 'sorted' at all times, you can write the key-file with fewer disc accesses, and with no overhead to key-split etc. This overhead is transferred to the reorganization procedure, which is run during nighttime, (maybe from a sleeper-process) when the load on the system is nil. The retrieval of the data does not take as many disc accesses, as you should expect, because, in the most dynamic files, a whole block of data is retrieved from one single key.

This structure, with reorganization of the file, forces you to split your data, according to how dynamic they are, to get the best performance. The data can be divided into three catagories :

- Activity data,	where you adds, modifies and deletes constantly.
- Update data,	where you modifies constantly, adds frequently and deletes seldomly.
- Static data,	where you modifies, adds and deletes seldomly.

The reorganization procedure removes all deleted entries in the data-file, and build a new key-file. As a special feature, you can specify some alternative keys, for ad hoc programs, monthly reports, or to establish a relation between different logical files.

Our expectations to the D-file

- Improved performance through fewer disc accesses
- Improved performance because key-file reorganization is running at night
- Improved up-time because no time consuming recovery is needed after a system failure
- Improved portability as all routines are written in FORTRAN

References :

"Systems Development, Projects Life - The practical experience"
Paper by Carl Christian Lassen at the IUG Edinburgh Conference 1983

"The Way To The Right Solution"
Article by Carl Christian Lassen, Supergroup Newsletter July 1982

"Centralized Database Access"
Paper by Erik Wendelboe at the IUG Copenhagen Conference 1982

"Solve Your Problems With Imagination"
Paper by Erik Wendelboe at the IUG Edinburgh Conference 1983

"Experiences With The SL Tecniques, Easing The Development
And Maintenance Burden"
Paper by Hanne Hansen at the IUG Copenhagen Conference 1983

SOFTWARE PROTOTYPING: TODAY'S APPROACH TO INFORMATION SYSTEMS DESIGN AND DEVELOPMENT

ORLAND LARSON
HEWLETT-PACKARD

Among the challenges facing the data processing community are the increasing costs and time associated with developing applications, the increasing backlog of applications, the excessive time spent maintaining applications, and the shortage of EDP professionals. In addition, systems implementation and functionality are impaired due to the lack of tools which involve end-users in the system development process.

Meeting these challenges requires a more progressive approach to applications development - one that is significantly different from traditional system development cycles. This approach is called SOFTWARE PROTOTYPING.

This paper defines software prototyping, identifies its major uses, reviews the step-by-step prototype development process, and discusses the resources and skills required to effectively prototype applications. It also addresses the problems and costs associated with software prototyping.

INTRODUCTION

The Changing Role of Data Processing

The data processing department has changed dramatically since the 1960's, when application development as well as production jobs were usually run in a batch environment with long turnaround times and out-of-date results.

The 1970's were a period of tremendous improvement for the data processing environment. One of the key developments of that period was the development and use of Data Base Management Systems (DBMS). This provided the basis for on line interactive applications. In addition, computers and operating systems provided programmers the capability of developing application programs on line, sitting at a terminal and interactively developing, compiling, and testing these applications. The end user was also provided with easy to use on-line inquiry facilities to allow them to access and report on data residing in their data bases. This took some of the load off the programmers and allowed them to concentrate on more complex problems.

During the 1980's, for the Data Base Administrator and MIS manager, we see increased importance and use of centralized data dictionaries or "centralized repositories of information about the corporate data resources." We also see simpler and more powerful report writers for the end user and business professional. For the programmer, we see the use of very high level transaction processing languages to reduce the amount of code required to develop applications. Finally, the tools have been developed to effectively do software prototyping which will provide benefits to the end user as well as the application programmer and analyst.

Throughout the Seventies and Eighties, information has become more accurate, reliable, and available, and the end user or business professional is becoming more involved in the application development process.

Challenges Facing MIS

The MIS manager's number one problem is the shortage of EDP specialists. A recent Computerworld article predicted that by 1990 there will be 1/3 of a programmer available for each computer delivered in this country. Software costs are also increasing because people costs are going up and because of the shortage of skilled EDP specialists. The typical MIS manager is experiencing an average of two to five years of application backlog. This doesn't include the "invisible backlog", the needed applications which aren't even requested because of the current known backlog. In addition, another problem facing MIS management is the limited centralized control of information resources.

The programmer/analyst is frustrated by the changeability of users' application requirements (the only thing constant in a user environment is change). A significant amount of programmers' time is spent changing and maintaining users' applications (as much as 60% of their time). Much of the code the programmer generates is the same type of routines such as error checking, formatting reports, reading files, checking error conditions, data validation, etc. This can become very monotonous or counter-productive for the programmer.

The end user or business professional is frustrated by the limited access to information needed to effectively do his/her day-to-day job. This is especially true for those users who know their company has spent a great deal of money on computer resources and haven't experienced the benefits. The user's business environment is changing dynamically and they feel MIS should keep up with these changes. MIS, on the other hand, is having a difficult time keeping up with these requests for application maintenance because of the backlog of applications and the shortage of EDP specialists. Once the user has "signed off" on an application, he is expected to live with it for awhile. He is frustrated when he requests what he thinks is a "simple change" and MIS takes weeks or months to make that change.

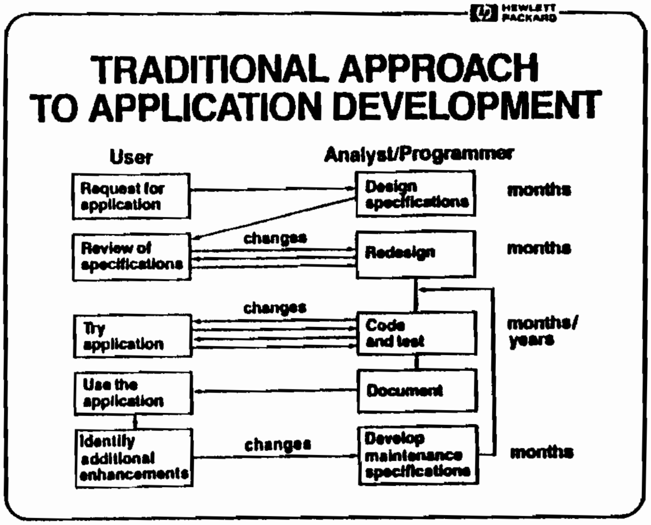
Traditional Approach to Application Development

There are some myths concerning application development:

- Users know what they want
- Users can communicate their needs to MIS
- Users needs are static

The traditional approach to application development has serious limitations when applied to on-line, interactive information systems that are in a state of constant change and growth. Communications among the user, analyst, programmer, and manager tend to be imprecise, a detailed analysis prolongs the process to the annoyance of the user, and specifications are either ambiguous or too voluminous to read. To compound this problem, the user is often requested to "freeze" his requirements and subsequent attempts at change are resisted.

Let's review the traditional approach to application development.



HEWLETT PACKARD

TRADITIONAL APPROACH TO APPLICATION DEVELOPMENT

User
Analyst/Programmer
Request for application
Design specifications
months
Review of specifications
changes
Redesign
months
Try application
changes
Code and test
months/ years
Use the application
Document
Identify additional enhancements
changes
Develop maintenance specifications
months

- The user first requests an application and then an analyst or programmer is assigned to the application.

- The analyst or programmer takes the oftentimes sketchy user specifications and designs more complete specifications.

- The user then reviews the analyst's interpretations of his specifications and probably makes additional changes.

- The analyst redesigns his specifications to adapt to these changes. (By this time, several days, weeks or months have gone by.)

- The user approves the specifications and a team of analysts and programmers are assigned to develop, test and document the application. (This may take months or years.)

- The user finally tries the application. Months or years may have gone by before the user gets his first look at the actual working application.

- The user, of course, will want additional changes or enhancements made to the application, to adjust the application to the "real world".

- Depending on the extent of these changes, additional maintenance specifications may have to be written and then coding, testing and documentation.

- The total application development process may take months or years and the maintenance of these applications may go on forever.

The question is: "Can MIS afford to continue using this traditional approach to application development?"

Prototyping Defined

According to Webster's Dictionary, the term prototype has three possible meanings:

1) It is an original or model on which something is patterned: an archetype.

2) A thing that exhibits the essential features of a later type.

3) A standard or typical example.

J. David Naumann and A. Milton Jenkins in a paper on software prototyping (see reference 3) believe that all three descriptions apply to systems development. Systems are developed as patterns or archetypes and are modified or enhanced for later distribution to multiple users. "A thing that exhibits the essential features of a later type" is the most appropriate definition because such prototypes are a first attempt at a design which generally is then extended and enhanced.

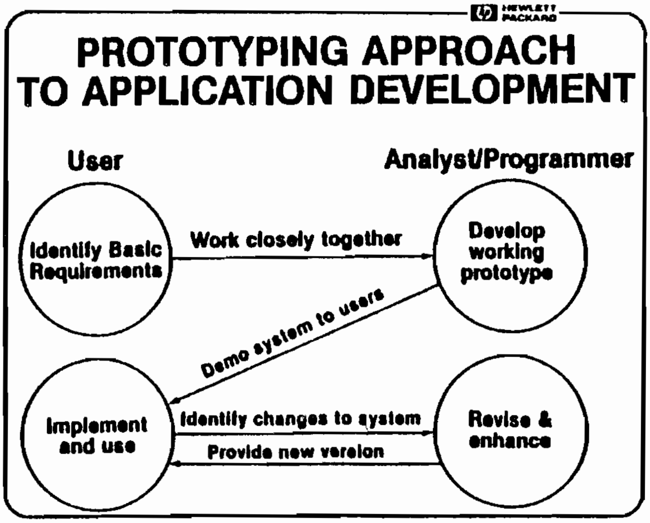
Software Prototypes

The process of software prototyping is a quick and relatively inexpensive process of developing and testing an application system. It involves the end user and programmer/analyst working closely to develop the application. It is a live, working system; it is not just an idea on paper. It performs actual work; it does not just simulate that work. It can be used to test out assumptions about users' requirements, system design, or perhaps even the logic of a program.

Prototyping is an iterative process. It begins with a simple prototype that performs only a few of the basic functions of a system. It is a trial and error process - build a version of the prototype, use it, evaluate it, then revise it or start over on a new version, and so on. Each version performs more of the desired functions and in an increasingly efficient manner. It may, in fact, become the actual production system. It is a technique that minimizes the dangers of a long formal analysis and increases the likelihood of a successful implementation.

The Prototype Model

Prototyping an information system can be viewed as a four step procedure.



PROTOTYPING APPROACH TO APPLICATION DEVELOPMENT

User
Analyst/Programmer
Identify Basic Requirements
Work closely together
Develop working prototype
Demo system to users
Implement and use
Identify changes to system
Provide new version
Revise & enhance

Step 1. Identify users' basic requirements:

- End user and programmer/analyst work closely together.
- Concentrate on users' most basic and essential requirements.
- Define data requirements, report formats, screens, and menus.
- Need not involve written specifications.
- For larger systems, a design team may need to spend a few weeks preparing a first-effort requirements document.

Step 2. Develop a working prototype:

- Programmer analyst takes the notes developed in the user discussions and quickly creates a working system.
- Designs and/or defines data base and loads subset of data.
- Makes use of defaults and standard report formats.
- Performs only the most important, identified functions.

Step 3. Implement and use the prototype:

- Programmer/analyst demonstrates prototype to small group of users.
- Users may request enhancements during demo.
- Users make notes of all changes they would like made.

Step 4. Revise and enhance the prototype:

- Programmer/Analyst and user discuss desired changes.
- Changes and enhancements for the next version are prioritized.
- Programmer/Analyst creates next version.
- Go back to Step 3.

NOTE: Steps 3 and 4 are repeated until the system achieves the requirements of this small group of users. Then either introduce to a larger group of users for additional requirements or if enough users are satisfied, demo to management to gain approval for the production system.

Uses of Software Prototypes

1. To clarify user requirements:

   - Written specs are often incomplete, confusing, and take a static view of requirements.
   - It is difficult for an end user to visualize the eventual system, or to describe their current requirements.
   - It is easier to evaluate a prototype than written specifications.
   - Prototyping allows - even encourages users to change their minds.
   - It shortens the development cycle and eliminates most design errors.
   - It results in less enhancement maintenance and can be used to test out the effects of future changes and enhancements.

2. To verify the feasibility of design:

   - The performance of the application can be determined more easily.
   - The prototype can be used to verify results of a production system.
   - The prototype can be created on a minicomputer and then that software prototype may become the specifications for that application which may be developed on a larger mainframe computer.

3. To create a final system:

   - Part (or all) of the final version of the prototype may become the production version.
   - It is easier to make enhancements and some parts may be recoded in another language to improve efficiency or functionality.

Essential Resources

The following are the essential resources to effectively do software prototyping:

1. Interactive Systems

   - Hardware and Operating System - When doing software prototyping, both the builder and the system must respond rapidly to the user's needs. Batch systems do not permit interaction and revision at a human pace. Hardware and associated operating systems tailored to on-line interactive development are ideal for software prototyping.

2. Data Management Systems

   - A Data Base Management System provides the tools for defining, creating, retrieving, manipulating, and controlling the information resources. Prototyping without a DBMS is inconceivable!

   - A Data Dictionary provides standardization of data and file locations and definitions, a cross reference of application programs, and a built-in documentation capability. These are essential to managing the corporate resources and extremely useful when prototyping.

3. Generalized Input and Output Software

   - Easy to use data entry, data editing, and screen formatting software are extremely helpful in the software prototyping process to allow the programmer to sit down at a terminal with a user and interactively create the user's screens or menus.

   - Powerful easy-to-use report writer and query languages provide a quick and effective way of retrieving and reporting on data in the system. A report writer that uses default formats from very brief specifications is most useful in the initial prototype.

4. Very High Level Languages

   - Traditional application development languages such as COBOL may not be well suited for software prototyping because of the amount of code that has to be written before the user sees any results.

   - Very powerful high level (MACRO) languages that interface directly to a data dictionary for their data definitions are ideal. One statement in this high level language could realistically replace 20-50 COBOL statements. This reduces the amount of code a programmer has to write and maintain and speeds up the development process.

5. Library of Reusable Code

   - A library of reusable code to reduce the amount of redundant code a programmer has to write is an important prototyping resource.

   - This code could represent commonly used routines made available to programmers.

Potential Problems

What are the problems with prototyping? How can data processing management control its use and keep it within bounds?

One problem with prototyping is the acceptance of this method by the systems people. It also may encourage the glossing over of the systems analysis portion of a project. It may be difficult to plan the resources to develop a system. Programmers may become bored after the nth iteration of the prototype. Testing may not be as thorough as desired and it might be difficult to keep documentation on the application up to date because it is so easy to change.

Even with these concerns, prototyping provides a very productive user-designer working relationship. So it behooves all data processing executives to learn to use this powerful tool creatively and to manage it effectively.

The advantages of prototyping greatly outweigh the problems.

Cost and Efficiency

It has been found that there is an order of magnitude decrease in both development cost and time with the prototype model.

It is often difficult to estimate the cost of an application system because the total costs of development, including maintenance are usually lumped together. The cost of implementing the initial system is much lower than the traditional approach (typically less than 25%).

However, software prototyping could be expensive in three ways:

1. It requires the use of advanced hardware and software.
2. It requires the time of high level users and experienced designers.
3. Efficiency may be compromised.

The main thing to remember is that the main focus of prototyping is not so much efficiency but effectiveness.

Summary

Prototyping is truly a "state of the art" way of developing applications.

- Software prototyping promotes an interactive dialogue between the users and the programmer, which results in a system being developed more quickly, and results in an interactive development approach which is friendlier for the end user.

- The prototype provides a live working system for the users to experiment with instead of looking at lengthy specifications.

- The users are provided with an early visualization of the system which allows them to immediately use it.

- The users are allowed and even encouraged to change their minds about user interfaces and reports.

- Maintenance is viewed right from the beginning as a continuous process and because the prototype is usually written in a very high level language, changes are faster to locate and easier to make.

- Software prototyping results in:

  * Users who are much more satisfied and involved in the development process.
  * Systems that meet the user's requirements and are much more effective and useful.
  * Improved productivity for all those involved in software prototyping: the users, the analysts, and the programmers.

Hewlett-Packard's Prototyping Tools

Hewlett-Packard is one of the few vendors that supplies the majority of the tools needed to effectively do software prototyping.

- Interactive Systems
    - HP3000 (all series)
    - MPE Operating System
- Data Management Systems
    - IMAGE/3000
    - KSAM/3000
    - MPE files
    - DICTIONARY/3000
- Generalized Input/Output Software
    - VPLUS/3000
    - QUERY/3000
    - REPORT/3000
    - INFORM/3000
    - DSG/3000
- Very High Level Languages
    - TRANSACT/3000

Bibliography

Canning, Richard G., "Developing Systems By Prototyping," EDP Analyzer (19:9) Canning Publications, Inc., September, 1981.

Naumann, Justus D. and Jenkins, A. Milton, "Prototyping: The New Paradigm for Systems Development," MIS Quarterly, Vol. 6, No. 3, September 1982.

Naumann, Justus D., and Galletta, Dennis F., "Annotated Bibliography of Prototyping for Information Systems Development," Management Information Systems Research Center Working Paper (MISRC-WP-82-12), September 1982.

Note: The above working paper as well as the paper by Naumann and Jenkins entitled "Prototyping: The New Paradigm for Systems Development," MIS Research Center-Working Paper (MISRC-WP-82-03), October 1981, are available for $3.00 each from :

University of Minnesota Systems Research Center
School of Management
269 19th Avenue South
University of Minnesota
Minneapolis, Minnesota 55455

or by calling 612-373-7822.

Podolsky, Joseph L., "Horace Builds a Cycle," Datamation, November 1977, pp.162-186.

Title: IMAGE/3000 STRATEGY

Author: Wendi Brubaker

Position: Product Line Manager, (408) 725-8111
Ex. 3765

Address: 19420 Homestead Rd., Cupertno, CA 95014

Are you a HP employee? Yes

Abstract:

Virtually every HP3000 customer uses the IMAGE/3000 data base management system to help make everyday business decisions. Part of Hewlett-Packard's strategy is to protect that large software investment and to provide a growth path for the future.

In this calendar year, HP will be announcing several IMAGE/3000 enhancements which focus on improved recovery systems and larger applications. Intrinsic Level Recovery will guarantee the physical integrity of a data base after a system crash. Logical integrity will be improved through an enhancement to the DBRecov utility. This will allow the log file to be quickly scanned for transaction begin and end pairs. Incomplete transactions against a single data base will be backed out when the affected data base is reopened.

1983 also promises MPE-V with disc caching which, with additional memory, will improve IMAGE/3000's performance on the Series 4X and 6X computers.

In the future, HP plans to provide a next generation IMAGE/3000 II product on HP3000's with MPE-V. IMAGE/3000 II will elevate the current data base size limits and increase the maximum number of concurrent users. There will also be an optional external interface to IMAGE/3000 II which will conform to the emerging HP IMAGE standard. Over the next few years, HP IMAGE will be offered across a range of HP computers.

HP is committed to the future of IMAGE across the complete range of HP technical and business computers.

Alan T. Pare
Hewlett Packard

The Data Dictionary: An Emerging System Resource

The utility of the data dictionary grew out of its first role as a documentation tool for managing data base definitions in the mid 1970's. The spectrum of its uses has evolved to a multitude of services for system data resources ranging from control, standardization, reporting, documenting, to aiding in the development of application programs. The way it is employed often depends on the organization's requirements and obviously the data dictionary system (DDS) capabilities itself. Today, information management necessitates stricter control to allow timeliness in auditing practices, security enforcement, analyzing the impact of system changes, maintenance procedures, design, and other operations tied to this objective. Cost effectiveness is paramount now more so than ever with higher labor expenses and vast amounts of complex system data and application programs to be developed and maintained. The DDS facilitates all of these needs by providing a centralized repository for documenting attributes, relationships, location, transactions, ownership, and various other data about data as referenced in the system environment. To support this array of functions, the DDS is being integrated with the operating system, program development environment, language compilers, report writers, query facilities, and other subsystems to play a more active role in the execution process of data access. This entails a high degree of sophistication and intelligence to be built into the DDS and direct interfaces with all referencing services. Optimizing performance in accessing the data dictionary, providing independence from physical storage structure for its dependent components, and customizing the DDS around an organization's requirements introduces even a higher degree of complexity in its implementation.

The DDS technology is more understood today with the advent of numerous research projects, technical literature and text books, and current product offerings particularly those on large mainframes. In the future, the DDS will be extended to provide more accurate ways of modeling the system environment, prototyping and planning more effectively, aid in developing friendlier end user products, and support distributed processing functions. The widespread proliferation of DBMS usage is placing additional burdens on the DDS to better support the DBA's responsibilities. The data dictionary system will eventually permeate the system architecture to optimize its controlling ability and performance. Figure 1 gives an overview of what a DDS of the future might look like.

Realizing the needs of its customer base and the importance of this flourishing technology, Hewlett-Packard is agressively moving to meet this challenge.The corporate research labs are studying advanced areas of DDSs dealing with information oriented properties of data related to the semantics of system data bases, i.e. real world entities as represented by data objects and their meaningful properties. The development lab is pursuing investigations into an integrated system wide DDS with a view toward distributed processing.

While these activities reflect a committment to understanding the future implementation methodology required to develop a state of the art DDS on Hewlett-Packard computers, activities are currently underway to meet the needs of our growing customer base in this area today.

Figure 2 illustrates the DICTIONARY/3000 applications data dictionary and its supporting RAPID/3000 products. Our short term strategy is now aimed at moving Dictionary/3000 into a more active role with manufacturing and financial applications, the program development environment, the language compilers, and a host of supporting utilities for VPLUS/3000, IMAGE/3000, multiple dictionaries, and future data base products. Figure 3 depicts the projects currently under development to achieve this integration.

The integration of MM, PM, and FA/3000 with DICTIONARY/3000 will allow the loading of information about their respective application's data bases and files into the DICTIONARY/3000 dictionary. This process will be a user option configured through the customizer and activated via the SAI terminal during preparation of the MM, PM, or FA/3000 applications dictionary. During this operation a job will be spawned to move the appropriate data definitions and relationships into the DICTIONARY/3000 dictionary. This process will be free of user interaction and may be reactivated at a later time if any of the MM, PM, or FA/3000 data bases or files should be modified. This new feature will provide customers using these manufacturing and financial applications packages the opportunity to use the report writing or applications language products offered by RAPID/3000.

For applications development in languages other than TRANSACT/3000, source code generation will be supported for PASCAL/3000 and COBOL/3000 in the HPTOOLSET environment and by means of a standalone utility. For COBOL/3000 development, both data and environment divisions can be generated and placed either in the HPTOOLSET edit files or COBOL/3000 copylibs with full editing capabilities being supported. In the standalone utility the same functionality will be offered except the COBOL/3000 copylib facility must be used for copylib editing. For PASCAL/3000, source code will be generated for both VAR and TYPE declarations in both the HPTOOLSET and standalone utility environments.The source code generated for both languages will optionally generate special source code for VPLUS/3000 and IMAGE/3000 variable names. In the standalone utility, input commands can be directed to an output file for subsequent use in batch mode or interactive updating of program declarations in the event program referenced variables definitions or relationships should change. Special features for default naming and data mapping will also be provided for both the HPTOOLSET and standalone utility user. This source code generation feature will provide a powerful mechanism for improving productivity in application program development, maintenance, and control.

The DICTIONARY/3000 utilities will be enhanced to support the loading of existing VPLUS/3000 forms files definitions into the data dictionary. The merging of one DICTIONARY/3000 dictionary into another will be supported. End user security will be derived from DICTIONARY/3000 to provide more transparency in such products as INFORM/3000. Future product releases in the area of relational data base technology will also be addressed.

In summary, it is the goal of Hewlett-Packard to respond in an expedient manner to its customer base needs and future system requirements in this important and revolutionary area of information management technology.

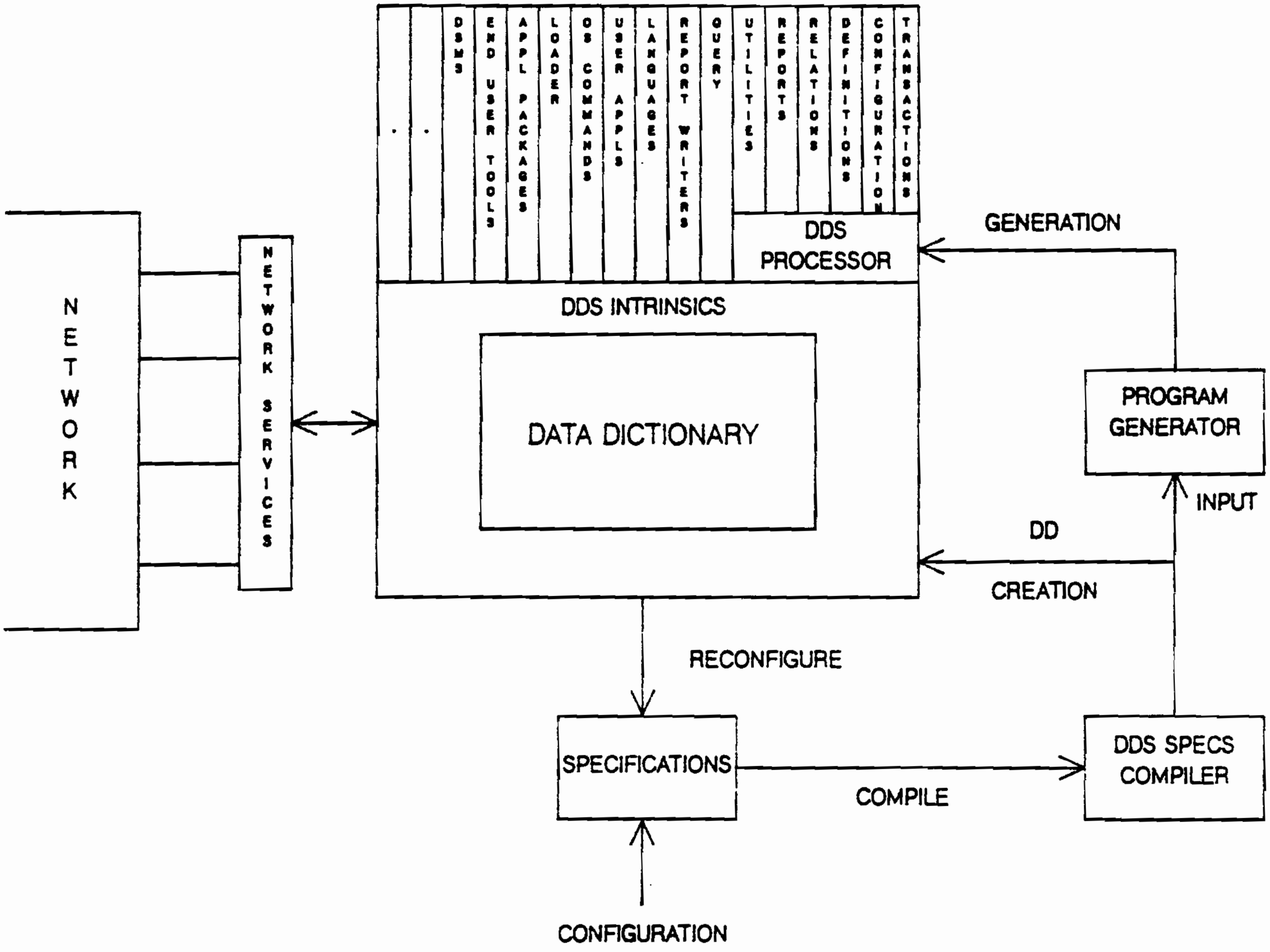

TRANSACTIONS
CONFIGURATION
DEFINITIONS
RELATIONS
REPORTS
UTILITIES
QUERY
REPORT WRITERS
LANGUAGES
USER APPLS
OS COMMANDS
LOADER
APPL PACKAGES
END USER TOOLS
DBMS
DDS PROCESSOR
GENERATION
PROGRAM GENERATOR
INPUT
DD
CREATION
DDS INTRINSICS
DATA DICTIONARY
NETWORK SERVICES
NETWORK
RECONFIGURE
SPECIFICATIONS
COMPILE
DDS SPECS COMPILER
CONFIGURATION

FIGURE 1 DDS ENVIRONMENT OF THE FUTURE

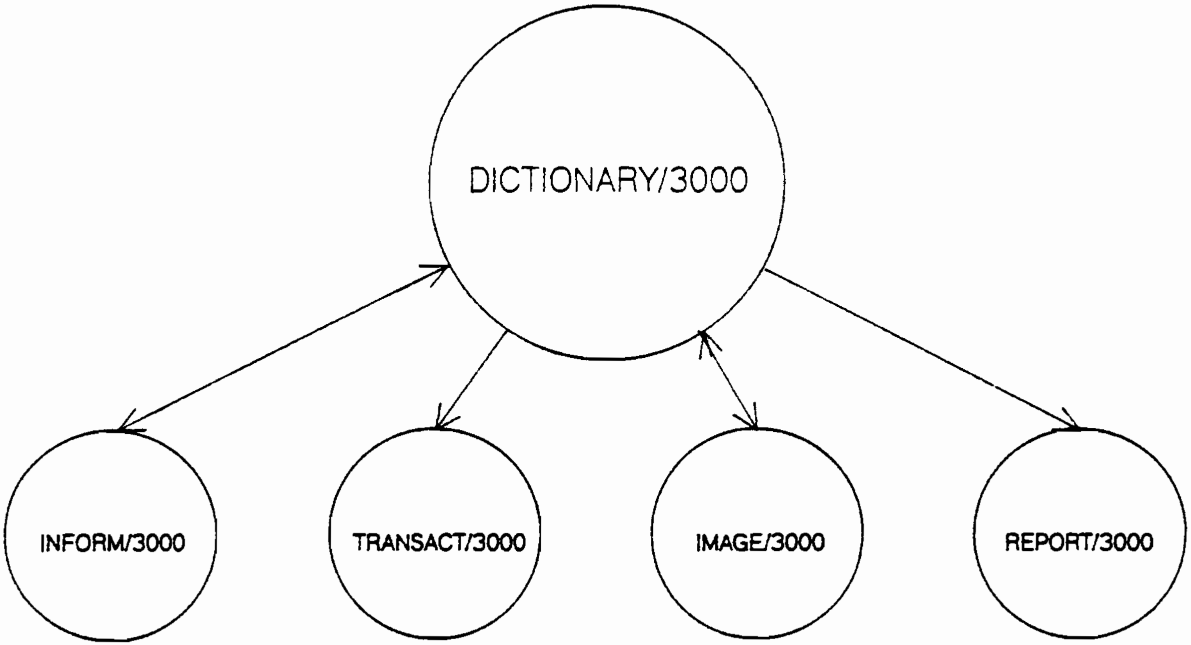

DICTIONARY/3000
INFORM/3000
TRANSACT/3000
IMAGE/3000
REPORT/3000

FIGURE 2 DICTIONARY/3000 ENVIRONMENT TODAY

Mike Kosolchroen
Hewlett Packard

Customizable Software - Why You Need It

User friendliness, or ease of use, has been widely heralded as the most significant attribute of a software package. However, another important attribute is the ability to accommodate changes without interrupting the system.

A lot of time, resources, and system interruptions are involved when code is modified to accommodate the simple changes in data base attributes. For instance, a Hewlett-Packard division took five months and the efforts of several programmers to make data base changes on accounting codes and employee codes, because of the number of programs involved and the number of systems affected. Customizable software allows you to implement these changes without code alteration and in only a few hours.

The concept of customizable software is not new. However, very few systems have incorporated the techniques necessary to derive its full benefits. The spiraling cost of software maintenance, combined with the need for fast system modification to cope with the dynamic business environment, make customizable software the most expedient answer.

Hewlett-Packard has developed two special software utilities called the Application Customizer and the Application Monitor. These utilities are incorporated in two manufacturing systems - HP Materials Management/3000 (MM/3000) and HP Production Management/3000 (PM/3000)- and a financial system. As a result, users can easily tailor these systems to their operations, without data processing staff. At the same time, HP continues to support the customized version of the software. This strategy has been well received, based on feedback from over 600 hundred installations.

This paper discusses the concept of software customization, the advantages and disadvantages, and the implementation of a customizable system. The HP system and its implementation are also reviewed in this paper.

Concepts of Software Customization

In general, software customization can be classified into four major areas:

A) Input or Screen Customization. This process involves modification the format of input or data entry screens. In addition to cosmetic changes, the content, field sequence, and field editing are redefinable as well. V/3000 is used in the HP system for this function.

B) Data Customization. This type of customization includes the ability to modify:
  - Data item characteristics, such as data type displayed numeric, packed, zoned, etc., data item length, data precision (number of decimals), the default or initial value, and even the naming of the data item.
  - Data structures. Fields can be added to or deleted from a data set or a file record layout.

C) Logic Customization. This involves altering the system processing logic via parameters, flags, or user-defined processing logic. This is the most difficult kind of customization, since logic is usually hard-coded.

D) Output or Report Customization. This aspect of customization involves changing the format of the output screen or the hard copy report. New fields can be included and existing fields can be removed from the report. The entire report, both the heading and content, are transformable.

Remember that because these areas of customization do not affect the basic programs, code does not have to be modified.

Example of Application System Customization

One example of application customization is modifying an inventory count transaction for an oil refining environment. In MM/3000, this transaction performs a warehouse inventory level count. In the standard package, the user enters the counted part number, the quantity, the warehouse, and the location ID. This transaction does not quite fit the refinery inventory count practice, since the refinery's parts are crude oil kept in giant oil tanks! One customer completely customized this standard count transaction. Rather than having to enter in units (in this case gallons), the system users enter only the height of the liquid in the tank, the ambient temperature, and the tank ID. With a set of front-end processing logic, the system calculates the volume of the tank using the tank's diameter (based on the tank ID.) and the tank's height, and adjusts for the coefficient of expansion using the ambient temperature. All these modifications were made through user-defined specifications, and no code was changed.

Advantages and Disadvantages of Customizable Software

Customizable software offers many benefits. This software is:

A) Inexpensive to operate. Customization does not require data processing staff, so this scarce resource is freed for other tasks.

B) Fast to implement changes. Since no programming effort is involved and customization is done via a special utility, modification can be made quickly and with minimum system interruption. The system can be designed to allow as much time as the user needs to redefine the application and its environment. Meanwhile, the application system continues functioning using the current specifications. Actual changes occur only when the users are ready to implement them.

C) Reliable. The traditional method of code modification is not only time-consuming, but also problem-prone. In customizable software, the application programs are separated and remain unchanged. The chance of introducing new bugs via customization is almost nil. Customization also allows much shorter testing time than the code change approach.

D) Versatile. Versatility is unquestionably one of the biggest advantages. This versatility gives the users freedom to tailor systems to meet their specific needs. There are many success stories from MM/3000 customers using the Application Customizer. One customer has completely transformed this material management package, which is an object code product, to a facility maintenance management system. Other customers have tailored the standard package to serve their refinery operation, which is a continous flow environment.

There are also certain disadvantages associated with customizable software:

A) Developing customizable software is not a simple task. A lot of time must be invested in the development stages. Several hundred thousand lines of code were created to implement customization in HP software. This may not be financially feasible for smaller EDP shops. At HP, this concept is economical because of the large number of packages sold.

B) Performance impact. Unfortunately, nothing comes free. Customizable software, in theory, will cost users more in terms of performance than the non-customizable version. A poorly-designed system can suffer substantial performance degradation. However, improved hardware price/perfor-mance, coupled with sophisticated software technology and appropriate choice of the implementing language, should minimize the

> performance impact. The HP system was implemented in SPL (HP/3000 Systems Programming Language) and employed the most current software technology.

It is important to realize that there is, of course, a limit to the extent of modification. The basic processing logic must be capable or suitable for the application.

Hewlett-Packard Implementation

HP's strategy is to circumvent the need for the application programs to handle data directly. The Application Customizer was designed to perform this task. Its key component is a special data base called the Application Data Dictionary. This data base maintains all the user-definable application parameters. Information such as data base schemas, data item attributes, screen and report formats, terminal configurations, and security are preserved. Figure 1 describes the contents of the Application Data Dictionary in more detail.

The Application Customizer provides an on-line, menu-driven facility for users to customize the system via the user-update version of the Application Data Dictionary. This data dictionary is later transformed, by the Application Customizer, into a more efficient run-time version. With two versions of the data dictionary, users can customize a system without affecting system operations, and then implement the customization when it is complete.

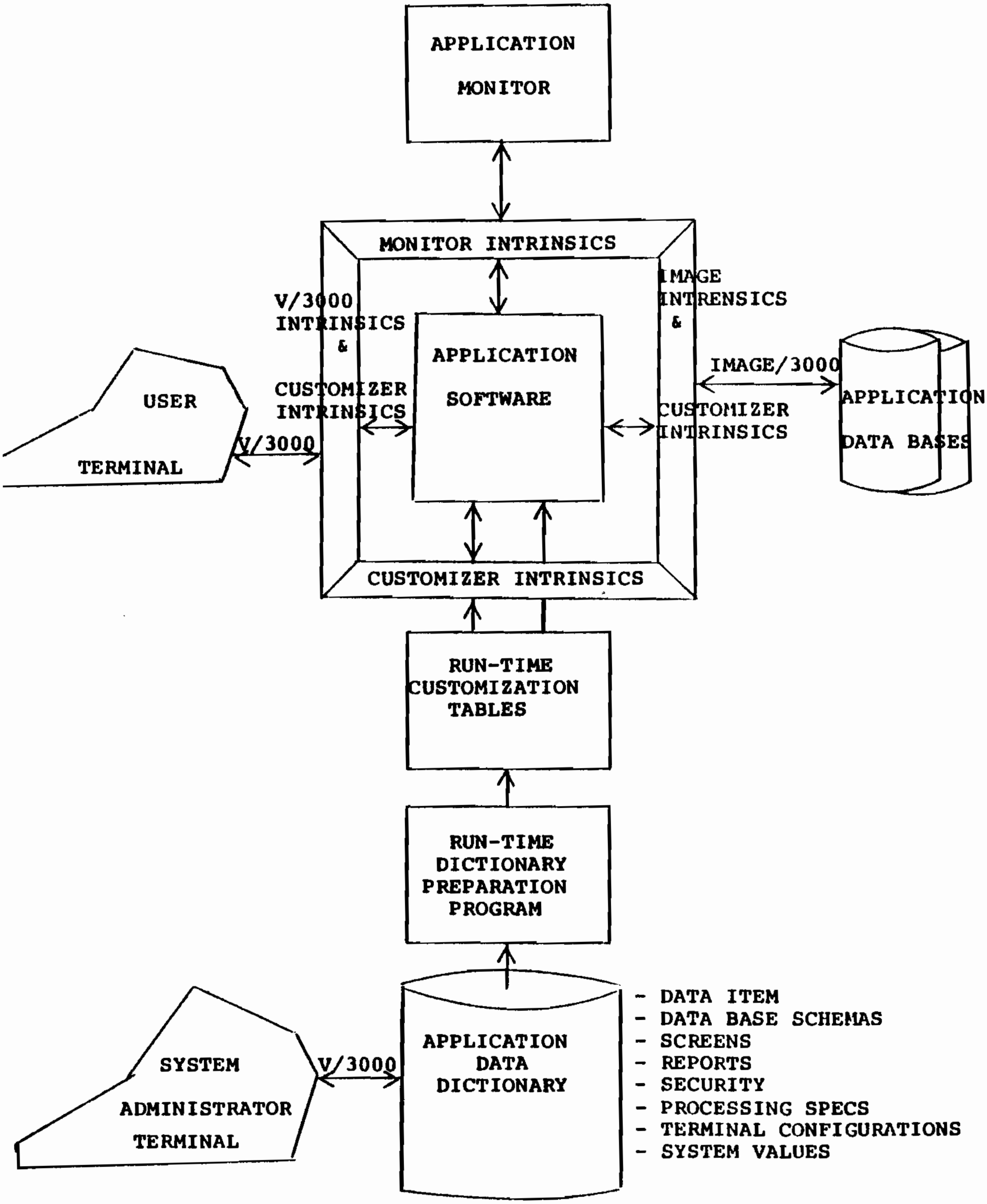

APPLICATION MONITOR
MONITOR INTRINSICS
V/3000 INTRINSICS & CUSTOMIZER INTRINSICS
IMAGE INTRENSICS &
APPLICATION SOFTWARE
USER TERMINAL
V/3000
IMAGE/3000
CUSTOMIZER INTRINSICS
APPLICATION DATA BASES
CUSTOMIZER INTRINSICS
RUN-TIME CUSTOMIZATION TABLES
RUN-TIME DICTIONARY PREPARATION PROGRAM
SYSTEM ADMINISTRATOR TERMINAL
V/3000
APPLICATION DATA DICTIONARY
- DATA ITEM
- DATA BASE SCHEMAS
- SCREENS
- REPORTS
- SECURITY
- PROCESSING SPECS
- TERMINAL CONFIGURATIONS
- SYSTEM VALUES

FIG. 1 The application programs are well insulated from other Input/Output functions. Communication to the other sub systems is conducted through intrinsics. The Application Data Dictionary contains all the data attributes and the application environment which are retrieved by the customizer intrinsics

A set of Application Customizer routines called customizer intrinsics were developed to handle all the arithmetic operations, data comparisons, data conversions, and data movements. Operations such as add, subtract, multiply, or divide are not performed directly in the application program, but are executed by calling the intrinsics. The intrinsics, in turn, reference the run-time data dictionary for necessary information, which may be data related or processing-logic-related. This design insulates the application programs from possible changes introduced by the system users.

There are many ways to accommodate logic customization. Many functions in the application programs can be performed using various logic choices specified via parameters called System Values. These algorithms are provided in anticipation of needs. A set of user-defined functions called processing specifications can also be entered via the Application Customizer. This instruction set, maintained in the Application Data Dictionary, allows simple data manipulation after normal processing but before the data base update. For most extensive logic alteration, where complicated data manipulation is needed, HP offers a special utility called User Exit. User Exit allows users to supplement the processing logic with their own programs. In general, the users' programs can be called at three major points: right after the screen input, before the data base update, and after the data base update. This latter provision is the ultimate solution to the customization of processing logic.

The system users may want some information, generated in one process, to be carried over and used or reported by another process. For example, users may want certain part attributes which are maintained in the Materials Management/3000 system to be transferred and reported in the Production Management/3000 system. To accommodate such a requirement, the record layout of the transfer file must be modified to include the new fields. In addition, the writing and reading programs must be modified. The Application Customizer offers a special type of format called the Data Exchange format. Using this type of format, application programs automatically associate each data item in the format with the corresponding field in the screen and the data set. Hence, the system users can define a new field and have the data updated or transferred to other processes or systems.

Traditionally, whenever modification is made to a file record layout, the associated job control file must also be modified, and MPE "BUILD" statement may need alteration on the record length. Customizable software should not require job control file modification whenever the file characteristics change. To meet this objective, a special command called FBUILD is used in lieu of the BUILD command. This command does not hardcode the record length. It specifies the record layout by referencing a format number corresponding to the file to be built. This technique allows file characteristics to change with no impact on the job control files. An example of an FBUILD command comparable to MPE'S BUILD command is shown in Figure 2 below.

L'AIR LIQUIDE

D.S.I.O./S.E.I.
JPB/GO

METHODS AND TOOLS

SUMMARY

I- / METHODS /

I-1) The concept of method p.1
I-2) System development method p.1

II- / TOOLS /

II-1) Preliminary study p.4
II-2) Detailed study p.4
II-3) Realization p.4
II-4) Implementation p.4

III- / METHOD AND TOOLS /

III-1) The problems ; primary objectives p.5
III-2) Initial developments p.5
III-3) Further developments p.11

I- / METHODS /

I-1) the concept of method

The concept of method in data processing covers many fields. We can group them into two areas.

- Project planning method concerns :

  . Project management
  . Development planning
  . Resources management
  . etc ...

- System development method proposes :

  . a philosophy for looking at problems
  . a guide for analysis, using concepts

Each of these two groups of methods has the some objectives :

- Increasing quality
- Increasing productivity
- Controlling costs and dead-lines.

We will try to show how, at L'AIR LIQUIDE, we tried to integrate methods and HP-tools use.

We will not say much about planning methods, because for this we use tools which are not implemented on HP computers.

I-2) System development method

The method we use is the MERISE method, which we have to describe briefly, if what follows is to be understood.

Philosophy

It consists in identifying several choice levels and two axes of analysis :

AXES LEVEL	DATA	PROCESSING
1	CONCEPTUAL	CONCEPTUAL
2	LOGICAL	ORGANIZATIONAL
3	PHYSICAL	OPERATIONAL

- Two axes : the important idea is that the two axes must be independant. The field of data is a data set, which is managed in the firm. It has its own existence, which does not depend on processing.

  Conversely, processes are the result of organizational choices, and do not depend on the existence of data.

- There are 3 levels of choice, as follows :

  . Management choices, for example :

    admit that an order can be delivered in one or several deliveries,

  . Organizational choices, for example :

    to execute part of an application in conversational, and leave another part entirely in manual,

  . Technical choices, for example :

    Install at a specific location a specific type of terminal.

They form successive strata from the more general to the more detailed, each level integrating the choices of the upper level. Those levels present a decreasing invariance degree, from conceptual to operational (or physical).

The steps

Four steps are identified :

- The preliminary study must lead to a definition of general policy and orientations within the guidelines of management and organization.

- The detailed study concerns the definition of detailed "external" fonctional specifications, and must end with user's agreement.

- Realization : the end result is a system in working order, after being subjected to the user's test data.

- Implementation in the system's real environment, and functioning at cruising spead for its final acceptance.

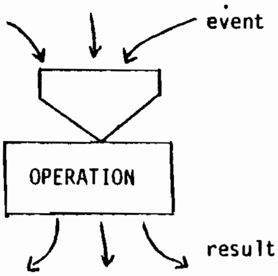
The manipulated concepts are the following :

- For the conceptuel data model ;
  . Objects (main entities such as "Product", "customer", ...)
  . Relations, such as "a customer takes an order for a Product", ...)

- For the logical data model ;
  . Records ⎱ concepts of CODASYL
  . Sets ⎰ formalization

- For the physical data model ;
  . Databases and/or files depending on the technical tools being used.

- For the conceptual processing model ;
  . The formalization is a classic one



- For the organizational processing model ;

  The same formalization is used with fonctional procedure instead of operation. A procedure implies the involvement of a work-station.

- For the operational processing model ;
  The procedures are described in sets of tasks.

The use of such a method enforces rigorous work. One of the disadvantages of this way of working is the need to write and then manage a lot paper in building the project's documentation. It also costs a lot of time.

Documentation management, for new system developments as for maintenance is a difficult problem.

This fact, though common, led us to consider that the method to be used must rely on automatization tools.

The choice for HP 3000 computers, at L'AIR LIQUIDE, (the first one was implemented in July 1982) was decided because of the interest of RAPID tools, and especially DICTIONARY.

II- / TOOLS /

The tools we use at L'AIR LIQUIDE on HP 3000 computers are classical tools one the one hand ;

- IMAGE, FCOPY, EDITOR, VIEW, ...
and RAPID tools on the other ;

- DICTIONARY, TRANSACT, INFORM, REPORT

When we began, we used them during the different steps as follows.

II-1) Preliminary study

No use.

II-2) Detailed study

Data

Introduction of the elements and their definition in DICTIONARY
Introduction of the physical model in DICTIONARY

Processing

Introduction of a short description of functional procedures in DICTIONALY.
Description of maps in FORMSPEC.

II-3) Realization

Standard use of TRANSACT, INFORM, REPORT.

II-4) Implementation

No use.

So we can use a chart to represent the rough contribution of the RAPID-tools in the following areas.

- DP DOC ; Documentation for DP people
- USER DOC ; Documentation for users
- CONC HELP; Help in conception for DP people.

		DP Doc	User Doc	Conc. Help	Concerned Levels		
					Concept	Organiz.	Operat.
Preliminary study		-	-	-			
Detailed Study	Data	XX	X	-			YES
	Process	XX	X	-			YES
Realization		X					YES
Implementation		-	-	-			

III- / METHOD AND TOOLS /

III-1) The problems ; primary objectives

Our first experience of HP 3000 project development led us quickly to several conclusions :

- Some of the steps find no contribution in using our tools,
- The contribution concerns only the operational level (no contribution to conceptual or organizational levels),
- The aid to documentation management is tangible for DP people, but slight for end-users,
- The use of these tools had to be defined with strict rules. Without these rules, there was a risk of not increasing productivity.
  Indeed, the freedom of use, for instance in TRANSACT programming, could lead to very dissimilar programs, and so to heavy and difficult maintenance.

Thus, it was clear that standard and RAPID-tools are interesting tools, but that they must be integrated with the MERISE-method and completed with accurate rules and specific tools.

In the face of urgent felt needs, the primary objectives were identified as :

- Rules for Dictionary use ; item-names coding,
- Standardization of screens and PF Keys,
- Standardization of TRANSAC procedures,
- etc ...

These first points are detailed as follows.

III-2) Primary developements

III-2.1) Item-names coding

A list of short-names was established first, and then completed progressively. The item-names are composed with these short-names.

For instance :

"NO-CLI" for 'NUMERO DE CLEINT'
"D-TAR" for 'DATE DE TARIF'

This standardization allows us the use of a common language on all projects. The consequent advantages are numerous.

Procedures are simple to read ; DP people can easily move from one project to another ; Maintenance becomes easier.

The short-names are inserted in DICTIONARY with a prefix ' * ' (* No, * CLI, * D, * TAR) so as to avoid conflicts with item-names. Their management is simple.

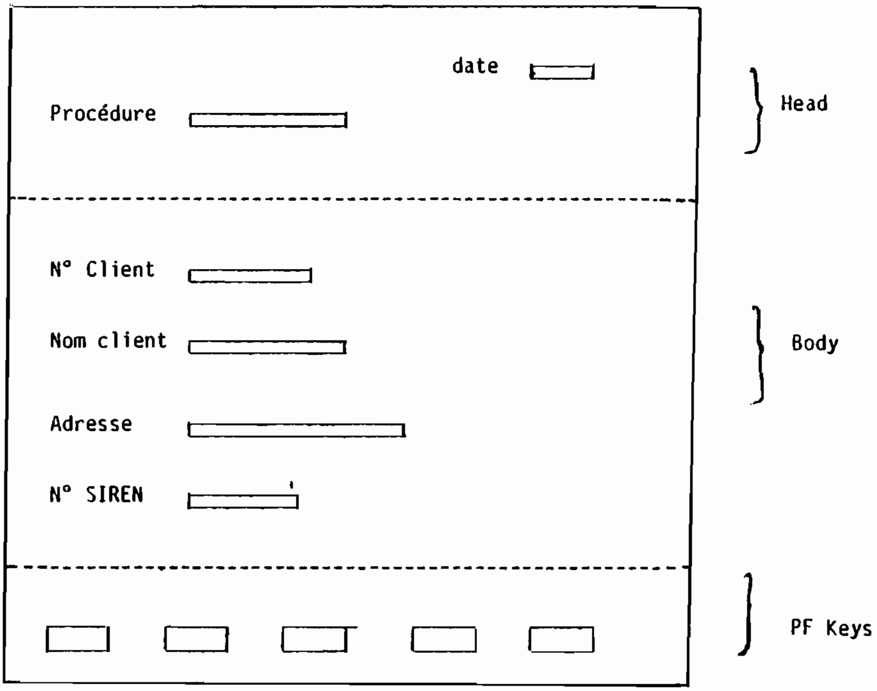
III-2.2) Standardization of screens

We classified the screens depending on their type and their character.

Screen type

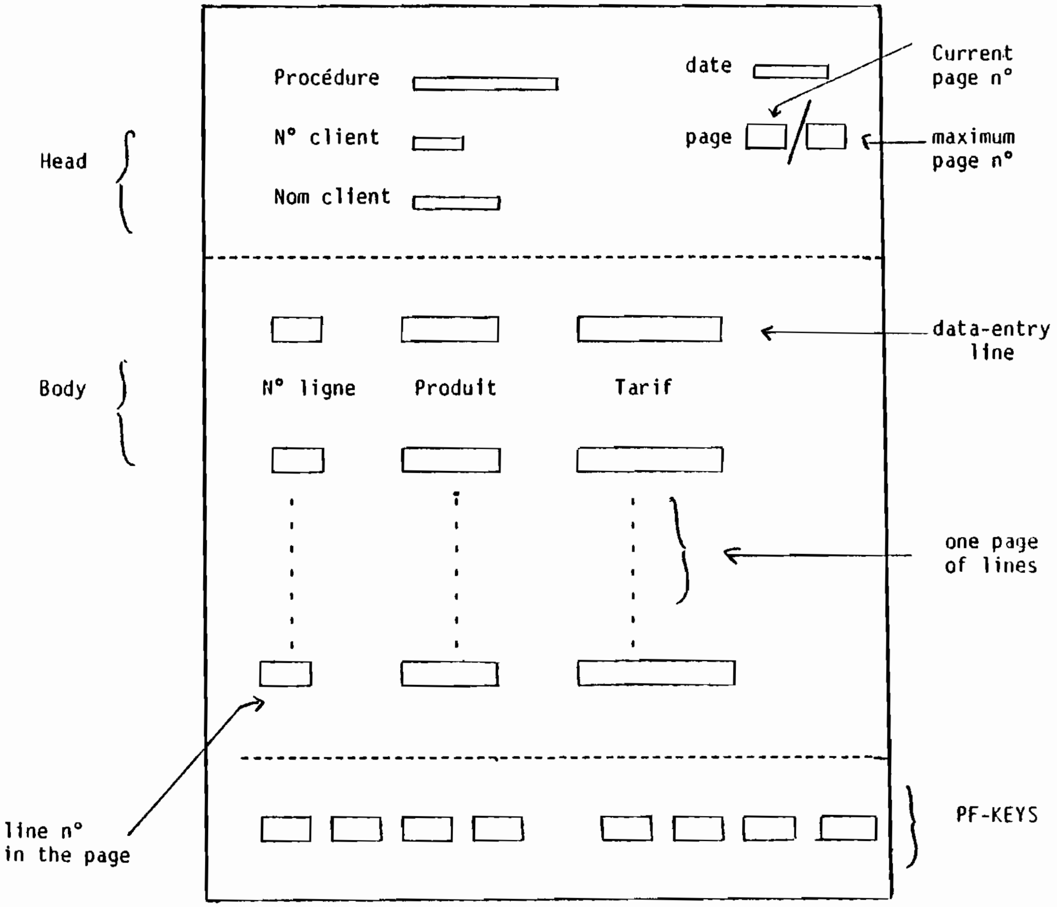
They are two types

Vertical screens present as following :

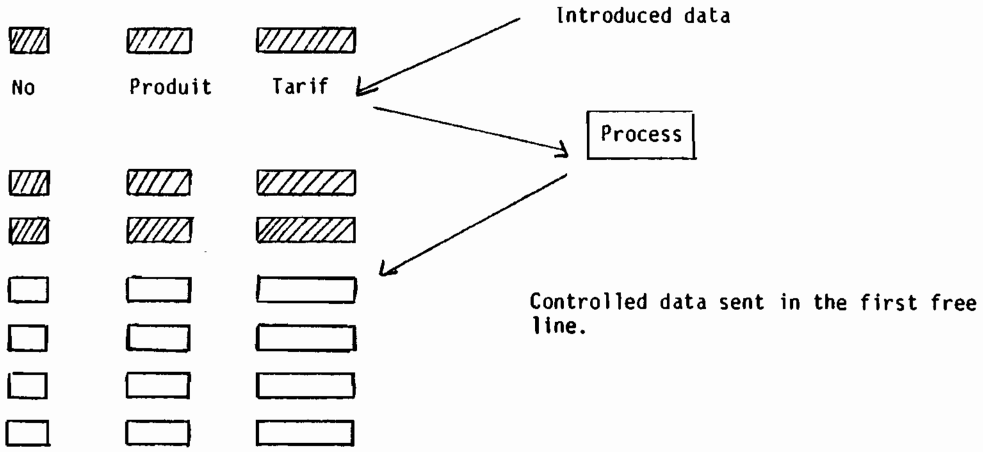

date
Procédure
Head
N° Client
Nom client
Adresse
N° SIREN
Body
PF Keys

The fields are not numerous and aligned on a column. Each field represents a singular item (no repeat items).

Horizontal screens present as following.


Procédure
date
Current page n°
N° client
page
maximum page n°
Head
Nom client
data-entry line
Body
N° ligne
Produit
Tarif
one page of lines
line n° in the page
PF-KEYS

The fields of the body-part are entered many times. Each time, they are controlled (formspec-process and/or transact-process), and moved, if correct, in the display-window (i.e. the display-page).


Introduced data
No
Produit
Tarif
Process
Controlled data sent in the first free line.

Moving from a page to another is obtained with the PF-Keys.

Screen character

We distinguish between

- creation between
- modification screens

and -querry screens

The consequences for screens-presentation is reduced to horizontal screens (there is no data entry-line on such a screen). But the difference is important for procedure-structure (see further).

Standardization of PF-Keys

We assigned standard use to the keys :

For instance Key 1 : confirm (in case of add or modify)

Key 2 : preceding page (in case of horizontal screen)

etc ...

Comments

This choice was dictated by concern for user-comfort.

The disadvantage is a heavy description for horizontal screens in FORMSPEC ; although the processing specifications concern only the "data-entry line", so efficiency remains good.

We also defined rules for dialog design, but it would be too lengthy to describe them here.

Advantages

These definitions allow us to :

- propose to end-users a pleasant and steady way of working
- propose to DP people working habits, so as to increase their mobility and minimize future programming loads.

Standardization of TRANSACT, TRAGEN generator

a) Technical principles

The characteristics of the program to be written are introduced (at the terminal). The result is a Transact program skeleton, stored in a MPE file (Editor-format). To become a source, it must be completed with specific code (with editor). It is then compiled in the normal way.

b) Characteristics

The programmer gives :

- program's code and name.
- Formspec file's name and quantity of screens

He gives then :

- the names of bases to be used

and defines, for each screen :

- Type and character
- Formspec name and literal name
- Fields description
- Standard keys to be used

In the case of a horizontal map, he has to define separately :

- the head fields
- the body-fields, with the number of lines in a page, and the number of pages to store in memory.

d) Skeleton/Specific code

The resulting skeleton includes :

- global declaration (SYSTEM order)

and for each map :

- DEFINE orders
- INIT orders
- Display and read orders for the map
- User's key declaration and process for standard keys
- Management for horizontal maps.

The programmer is so freed from the heavy and repetitive part of his work. He has to complete with "intelligent" code, such as :

- file-access
- external controls (i.e. non FORMSPEC-controls).

e) Observed results

The TRAGEN elaboration cost two months to write it (it is written in TRANSPACT). and two months to test before it worked efficiently.

Nowadays the skeleton represents 50 % to 80 % of the program (Number of lines), depending on the program's complexity.

f) Improvements

We intend to go further. It is possible to :

- Integrate file-access
- Integrate TRAGEN with DICTIONARY and FORMSPEC
- Integrate simple external controls.

g) Example

See in Annex an example with :

- TRAGEN declaration screens
- declared FORMSPEC screens
- final TRAGEN result.

III-3) Future developments

We can use other chart to represent some advantages observed, as a result of our first efforts ;

		OP Doc	User Doc	Conc. Help	Concerned Levels		
					Concept.	Organiz.	Operat.
Preliminary Study		-	-	-			
Detailed Study	Data	XX	X $\boxed{X}$	-			YES
	Process	XX	X	-			YES
Realization		X $\boxed{X}$	-	$\boxed{XX}$			YES
Implementation		-	-	-			

We intend to go further. New improvements could be obtained with new features for existing tools or with new tools. We have drawn up a list of these features and tools ; some are specific to our method, and we will of course have to develop them ourselves, but others are more general in nature, and we hope that future enbrancements of RAPID will include them. The list was established on the following simple principles :

- The Dictionary is the basis for most of the developements aimed at.
- Consequentely, no real progress is possible if this dictionary remains "closed". HP-users should be able to extend the dictionary, so as to manage new entities with it, depending on their own needs.
- Whatever method is used, it requires several steps, and the study of each step must be established on the basis of the preceding steps. Tools must make it possible to introduce these results progressively in the dictionary or connected areas, such as Formspec-files.
- the heavy parts of the development work should be automated as much as possible.

In our list, we also included the following features :

- We want to be able to describe in the dictionary new entities, such as "Object" or "Relation", so as to introduce the description of conceptual or organizational levels.
- We want more efficient tools concerning the print of Formspec-files, so as to use the reports for user-documentation. Moreover, it would be interesting for us to have much more flexibility using these tools, so as to adapt the reports to our needs, for instance in translating the processing specifications into french user-language ; or in generating Formspec-files from a shortened description according to our screens design-rules.

Such features cannot be developed without a contribution from HP.

Other features could be developed by ourselves, such as :

- Automation of user guides, based on the dictionary documentation and Formspec description. The user guides could be automatically displayed on the screen in response to a defined function key.
- Simple tools can be produced, as aids in conception, for instance conceptual data model conception.
- TRAGEN can easily be extended to include further automation.

These objectives are more or less ambitions, but we are sure that very important improvements are involved, concerning quality, efficiency, and documentation accuracy.

We estimated for instance some increases in efficiency, as follows.

- Conceptual data model :

  Dictionary extension is involved
  About 1 day is to earn in this way, for a 10 months project (200 days)
- Automation in screens-design : about 5 days (for the same 200 days-project)
- Increased automation of program-production (TRAGEN) :
  About 10 days.
- Automation in user guide production :
  About 20 days.

The total increase would then be almost 20 %.But, the improvements in quality and accuracy would have much more important effects.

- Important gains are possible by improving dialogue between DP people and users; this is difficult to estimate.

- The cost of maintenance could be reduced by more than 50 % through strict standardization and integrated documentation.

We can imagine the future chart, as follows :

		DP Doc	User Doc	Conc. Help	Concerned Levels		
					Concept.	Organiz.	Operat.
Preliminary Study		$\boxed{X}$		$\boxed{X}$	YES		
Detailed Study	Data	XX $\boxed{X}$	XX $\boxed{X}$	$\boxed{XX}$	YES	YES	YES
	Process	XX $\boxed{X}$	XX $\boxed{X}$	$\boxed{X}$	YES	YES	YES
Realization		XX $\boxed{X}$	not involved	XX $\boxed{X}$			YES
Implementation			$\boxed{XX}$	$\boxed{XX}$		YES	YES

In conclusion, we are sure that these objectives are not utopian. Someof them have already been reached on other machines and the RAPID-tools are a good basis for reaching them on HP 3000 computers.

```
L'AIR LIQUIDE
*************

                 AIDE A LA PROGRAMMATION TRANSACT



    Nom du Programme : 030101

              Libellé :            CREATION DE FICHE-EMBALLAGE

     Nombre d'écrans : 3
  Fichier des écrans : VLMS
```

```
L AIR LIQUIDE
*************

              AIDE A LA PROGRAMMATION TRANSACT


                   Ecrans utilis(s

Typ  Nat Nom abr(g(     Nom
Ecr  Ecr   Ecran        Ecran
 V    A  MCODE                         CODE-EMBALLAGE

 V    C  MDESCR                      DESCRIPTIF-EMBALLAGE

 H    C  MTAR                       TARIF GENERAL EMBALLAGE
```

```
L'AIR LIQUIDE
*************

              AIDE A LA PROGRAMMATION TRANSACT

                 Déclaration des Variables
       Ecran :      CODE-EMBALLAGE


    Nom de Variable     Typ Size

    C-EMB                X   04
```

```
L'AIR LIQUIDE
*************

               AIDE A LA PROGRAMMATION TRANSAC

          Ecran    CODE-EMBALLAGE


               Touches.Fonction  Standard    Ecran Suivant

             Touche 1 :   'Validation'
             Touche 2 :   'Page precedente'
             Touche 3 :   'Page suivante'
             Touche 4 :   'Effacement Ecran'
             Touche 5 :   'Ecran précedent'
             Touche 6 :   'Ecran suivant'
             Touche 7 :   'Copie Ecran'
             Touche 8 : S 'Abandon'

    Indiquez 's' lorsque vous souhaitez une utilisation standard
          ou donnez éventuellement le nom de l'Ecran Suivant
    Pour les Touches 5 et 6, et @pour elles seules, vous devez
          donner le nom de l'Ecran Suivant en cas d'utilisation Standard
```

```
L'AIR LIQUIDE
*************

          AIDE A LA PROGRAMMATION TRANSACT

             Déclaration des Variables :
   Ecran :    DESCRIPTIF-EMBALLAGE


      Nom de Variable    Typ Size

      C-EMB               X   04
      N-EMB               X   20
      C-PER-REC           X   01
      CONT-EMB            X   05
      U-CONT              X   01
      LONG-EMB            X   03
      LAR-EMB             X   03
      HAUT-EMB            X   03
      PDS-EMB             X   06
```

```
L'AIR LIQUIDE
*************

                AIDE A LA PROGRAMMATION TRANSAC

         Ecran     DESCRIPTIF  EMBALLAGE

                Touches.Fonction  Standard    Ecran Suivant

           Touche 1 : S 'Validation'
           Touche 2 :   'Page precedente'
           Touche 3 :   'Page suivante'
           Touche 4 : S 'Effacement Ecran'
           Touche 5 :   'Ecran pr{cedent'
           Touche 6 :   'Ecran suivant'
           Touche 7 : S 'Copie Ecran'
           Touche 8 : S 'Abandon'

      Indiquez 's' lorsque vous souhaitez une utilisation standard
          ou donnez Cventuellement le nom de l'Ecran Suivant
      Pour les Touches 5 et 6, et pour elles seules, vous devez
             donner le nom de l'Ecran Suivant en cas d'utilisation Standard
```

```
L'AIR LIQUIDE
*************

            AIDE A LA PROGRAMMATION TRANSACT

              D(claration des Variables :
     Ecran :     TARIF   GENERAL   EMBALLAGE


   Nom de Variable     Typ Size

   C-EMB                X   04

   N-EMB                X   20

L'AIR LIQUIDE


              AIDE A LA PROGRAMMATION TRANSACT
              D(claration des Tableaux utilis(s
                (Cas des Ecrans de Type 'H')

    Nombre maximum de Lignes Present(es @ l' Ecran : 08
       Nombre maximum de Pages Stock(es en M(moire : 05

   Nom de la Variable   Typ Siz

   D-EF-TAR              X   08

   NO-VRS-EMB            X   02

   PRX-U-VEN             X   06

   PRX-U-CSG             X   06

   PRX-U-ACH             X   06
```

```
IR LIQUIDE
**********

            AIDE A LA PROGRAMMATION TRANSAC

        Ecran - TARIF-GENERAL EMBALLAGE

            Touches Fonction Standard - Ecran Suivant

          Touche 1 : S 'Validation'
          Touche 2 : S 'Page_precedente'
          Touche 3 : S 'Page suivante'
          Touche 4 : S 'Effacement Ecran'
          Touche 5 :   'Ecran pr(cedent'
          Touche 6 :   'Ecran suivant'
          Touche 7 : S 'Copie Ecran'
          Touche 8 : S 'Abandon'

   Indiquez 's' lorsque vous souhaitez une utilisation standard
         ou donnez Cventuellement le nom de l'Ecran Suivant
   Pour les Touches 5 et 6, et @pour elles seules, vous devez
         donner le nom de l'Ecran Suivant en cas d'utilisation Standard
```

```
<<                                                                      >
<<                                                                      >
<<                         PROGRAMME : 030101                           >
<<                                                                      >
<<                    CREATION DE FICHE EMBALLAGE                       >
<<                                                                      >
<<                             3   ECRANS                               >
<<                                                                      >
<<                                                                      >

SYSTEM 030101,
       BASE=BDPROD.BASE.DEV1( "DC" , 1),
       VPLS=VREMB(MCODE(V-D-JOUR,V-LIB-PROC,V-C-EMB),
            MDESCR(V-D-JOUR,V-LIB-PROC,V-C-EMB,V-N-EMB,V-C-PER-REC,
            V-CONT-EMB,V-U-CONT,V-LONG-EMB,V-LARG-EMB,V-HAUT-EMB,
            V-PDS-EMB),
            MTAR(V-D-JOUR,V-LIB-PROC,V-C-EMB,V-N-EMB,CTR-PAGE-MTAR,
            PAGE-MAX-MTAR,NO-EC-MTAR,V-D-EF-TAR,V-NO-VRS-EMB,
            V-PRX-U-VEN,V-PRX-U-CSG,V-PRX-U-ACH,TAB-MTAR));


<<                                                                      >
<<                                                                      >
<<                            ECRAN NO 01                               >
<<                                                                      >
<<                           CODE EMBALLAGE                             >
<<                                                                      >
<<                                                                      >


MCODE-DEF-ITEM:

  DEFINE(ITEM) V-D-JOUR                 X(08);
  DEFINE(ITEM) V-LIB-PROC               X(50);
  DEFINE(ITEM) MESS-ERR                 X(78);
  DEFINE(ITEM) Z-ERR                    X(20);
  DEFINE(ITEM) V-C-EMB                  X(04);

MCODE-INIT-ITEM:

  RESET(STACK) LIST;
  LIST MESS-ERR,INIT:
       Z-ERR,INIT:
       V-D-JOUR,INIT:
       V-LIB-PROC,INIT:
       V-C-EMB,INIT;


MCODE-AFFICH:

  MOVE (V-LIB-PROC) = "                 CREATION FICHE EMBALLAGE";

  PUT(FORM) MCODE,INIT,LIST=(V-LIB-PROC),WINDOW=("");


MCODE-SAISIE:

  GET(FORM) MCODE, CURRENT,
     F0 = MCODE-SAISIE,
     F1 = MCODE-SAISIE,
     F2 = MCODE-SAISIE,
     F3 = MCODE-SAISIE,
     F4 = MCODE-SAISIE,
     F5 = MCODE-SAISIE,
     F6 = MCODE-SAISIE,
     F7 = MCODE-SAISIE,
     F8 = RETOUR-MENU;


<<
<<
<<                     ECRAN NO 02
<<
<<                  DESCRIPTIF-EMBALLAGE
<<
<<

MDESCR-DEF-ITEM:


  DEFINE(ITEM) V-N-EMB        X(20);
  DEFINE(ITEM) C-PER-REC      X(01);
  DEFINE(ITEM) V-CONT-EMB     X(05);
  DEFINE(ITEM) V-U-CONT       X(01);
  DEFINE(ITEM) V-LONG-EMB     X(03);
  DEFINE(ITEM) V-LARG-EMB     X(03);
  DEFINE(ITEM) V-HAUT-EMB     X(03);
  DEFINE(ITEM) V-PDS-EMB      X(06);

MDESCR-INIT-ITEM:

  RESET(STACK) LIST;
  LIST MESS-ERR,INIT:
       Z-ERR,INIT:
       V-D-JOUR,INIT:
       V-LIB-PROC,INIT:
       V-C-EMB,INIT:
       V-N-EMB,INIT:
       V-C-PER-REC,INIT:
       V-CONT-EMB,INIT:
       V-U-CONT,INIT:
       V-LONG-EMB,INIT:
       V-LARG-EMB,INIT:
       V-HAUT-EMB,INIT:
       V-PDS-EMB,INIT;


MDESCR-AFFICH:

  PUT(FORM) MDESCR,INIT,WINDOW=(" ");


M    CR-SAISIE:

     GET(FORM) MDESCR  CURRENT,
               F0 = MDESCR-CONTROLE,
               F1 = MDESCR-VALID,
               F2 = MDESCR-SAISIE,
               F3 = MDESCR-SAISIE,
               F4 = MDESCR-AFFICH,
               F5 = MDESCR-SAISIE,
               F6 = MDESCR-SAISIE,
               F7 = MDESCR-COPIE,
               F8 = ABANDON;


 MDESCR-EFFAC:


 MDESCR-COPIE:

     PROC HARDCOPY (VCOM(VEMB));
     GO TO MDESCR-AFFICH;


 MDESCR-CONTROLE:

 <<
 <<
 <<                                     ECRAN NO 03
 <<
 <<                           TARIF GENERAL DE L'EMBALLAGE
 <<
 <<


 MTAR-DEF-ITEM:

     DEFINE(ITEM) CTR-LG-MTAR             9(2);
     DEFINE(ITEM) LG-TRT-MTAR             9(2);

     DEFINE(ITEM) LIGNE-MTAR              X(0030):
                  NO-EC-MTAR              X(02)=LIGNE-MTAR(1):
                  V-D-EF-TAR              X(08)=LIGNE-MTAR(3):
                  V-NO-VRS-EMB            X(02)-LIGNE-MTAR(11):
                  V-PRX-U-VEN             X(06)=LIGNE-MTAR(13):
                  V-PRX-U-CSG             X(06)-LIGNE-MTAR(19):
                  V-PRX-U-ACH             X(06)=LIGNE-MTAR(25);

     DEFINE(ITEM) CTR-PAGE-MTAR           9(2);
     DEFINE(ITEM) PAGE-MAX-MTAR           9(2);

     DEFINE(ITEM) TAB-MTAR             08X(0039);

     DEFINE(ITEM) WTAB-MTAR            05X(0240):
                  WPAG-MTAR            08X(0030)  =  WTAB-MTAR  (1):
                  WPOS-MTAR               X(0039)  =  WPAG-MTAR  (1):
                  NO-POST-MTAR            X(02)  =  WPOS-MTAR  (0001):
                  W-D-EF-TAR              X(08)  =  WPOS-MTAR  (0003):
                  W-NO-VRS-EMB            X(02)  =  WPGS      R  (0011):
```

```
303.3                        W-PRX-U-VEN           X(06) = WPOS-MTAR (0013):
303.4                        W-PRX-U-CSG           X(06) = WPOS-MTAR (0019):
303.5                        W-PRX-U-ACH           X(06) = WPOS-MTAR (0025);
303.6
310       MTAR-INIT-ITEM:
310.1
310.2       RESET(STACK)=LIST;
310.3       LIST MESS-ERR,INIT:
310.4            Z-ERR,INIT:
310.5            LG-TRT-MTAR,INIT:
310.6            CTR-LG-MTAR,INIT:
310.7            V-D-JOUR,INIT:
310.8            V-LIB-PROG,INIT:
310.9            V-C-EMB,INIT:
311              V-N-EMB,INIT:
311.1            CTR-PAGE-MTAR,INIT:
311.2            PAGE-MAX-MTAR,INIT:
311.3            LIGNE-MTAR,INIT:
311.4            TAB-MTAR,INIT:
311.5            NTAB-MTAR,INIT;
311.6
311.7
311.8
311.9
340       MTAR-AFFICH:
340.1
340.2       PUT(FORM) MTAR,INIT,WINDOW=(" "),
340.3
340.4
345       MTAR-SAISIE:
345.1
345.2       GET(FORM) MTAR, CURRENT,
345.3            F0 = MTAR-CONTROLE,
345.4            F1 = MTAR-VALID,
345.5            F2 = MTAR-PAG-PREC,
345.6            F3 = MTAR-PAG-SUIV,
345.7            F4 = MTAR-SAISIE,
345.8            F5 = MTAR-SAISIE,
345.9            F6 = MTAR-SAISIE,
346              F7 = MTAR-COPIE,
346.1            F8 = ABANDON;
346.2
346.3
350       MTAR-PAG-PREC:
350.1
350.2       IF (CTR-PAGE-MTAR) = 1 THEN
350.3         DO
350.4           PUT(FORM) MTAR,
350.5             WINDOW=("PAS DE PAGE PRECEDENTE !"),
350.6             LIST=(V-D-JOUR:TAB-MTAR);
350.7           GO TO MTAR-SAISIE;
350.8         DOEND
350.9       ELSE
351           DO
351.1           LET (CTR-PAGE-MTAR) = (CTR-PAGE-MTAR) - 1;
351.2           LET OFFSET(WPAG-MTAR) = OFFSET(WPAG-MTAR)-0240;
351.3           MOVE (LIGNE-MTAR) = " ";
```

```
      MOVE (TAB-MTAR) = (WPAG-MTAR);
      LET (LG-TRT-MTAR) = 0;
      GO TO MTAR-AFFICH;
    DOEND;


MTAR-PAG-SUIV:

  IF (CTR-PAGE-MTAR) = (PAGE-MAX-MTAR) THEN
    DO
      PUT(FORM) MTAR,
        WINDOW=("PAS DE PAGE SUIVANTE !"),
        LIST=(V-D-JOUR:TAB-MTAR);
      GO TO MTAR-SAISIE;
    DOEND
  ELSE
    DO
      LET (CTR-PAGE-MTAR) = (CTR-PAGE-MTAR) + 1;
      LET OFFSET(WPAG-MTAR) = OFFSET(WPAG-MTAR)+0240;
      MOVE (LIGNE-MTAR) = " ";
      IF (CTR-PAGE-MTAR) = (PAGE-MAX-MTAR) THEN
        DO
          MOVE (NO-LC-MTAR) = (CTR-LG-MTAR);
          LET (LG-TRT-MTAR) = (CTR-LG-MTAR);
          LET OFFSET(WPOS-MTAR) = 0030*(LG-TRT-MTAR)-0030;
        DOEND
      ELSE
        DO
          LET (LG-TRT-MTAR) = 0;
        DOEND;
      MOVE (TAB-MTAR) = (WPAG-MTAR);
      GO TO MTAR-AFFICH;
    DOEND;


MTAR-EFFAC:

  MOVE (LIGNE-MTAR) = "-";
  MOVE (NO-LC-MTAR) = (LG-TRT-MTAR);
  GO TO MTAR-AFFICH;


MTAR-COPIE:

  PROC HARDCOPY (VCOM(VEMB));
  GO TO MTAR-AFFICH;


MTAR-CONTROLE:

<<                                        >>
<<      TEST POUR DISTINGUER :            >>
<<          UNE SAISIE EN SEQUENCE        >>
<<          UNE DEMANDE DE LIGNE          >>
<<          UNE MODIFICATION DE LIGNE     >>
<<                                        >>

```

```
 IF (NO-LC-MTAR) <> (LG-TRT-MTAR) THEN GO TO DDE-LG-MTAR;

<<                                                   >>
<<        CONTROLE DE LA LIGNE SAISIE                >>
<<        MISE EN TABLE APRES CONTROLE               >>
<<                                                   >>


  MOVE (NO-LC-MTAR) = " ";
  IF (LIGNE-MTAR) = " " THEN
   DO
    IF (CTR-PAGE-MTAR) <> (PAGE-MAX-MTAR) THEN
     GO TO SUP-LG-MTAR;
    IF (CTR-LG-MTAR) <> (NO-LC-MTAR) THEN
     GO TO SUP-LG-MTAR;
   DOEND;
  IF (LIGNE-MTAR) = " " THEN GO TO MTAR-SAISIE;
  MOVE (NO-LC-MTAR) = (LG-TRT-MTAR);

<<                                                >>
<<          CONTROLE DE LA LIGNE                  >>
<<                                                >>



SUP-LG-MTAR :
<<             SUPPRESSION DE LIGNE            >>
<<             SUPPRESSION DE LIGNE            >>
<<                                             >>
 MOVE (NO-LC-MTAR) = (LG-TRT-MTAR);


FIN-CONTROLE-MTAR:

 MOVE (WPOS-MTAR) = (LIGNE-MTAR);
 IF (CTR-PAGE-MTAR) <> (PAGE-MAX-MTAR) THEN GO TO FIN-MODIF-MTAR;
 IF (CTR-LG-MTAR) <> (NO-LC-MTAR) THEN GO TO REPRISE-SEQ-MTAR;

MTAR-INCREMENT-SEQUENCE :

 UNE-PAGE-MTAR :

  UN-POSTE-MTAR :

   LET OFFSET(WPOS-MTAR) = OFFSET(WPOS-MTAR) + 30;
   MOVE (LIGNE-MTAR) = " ";
   LET (CTR-LG-MTAR) = (CTR-LG-MTAR) + 1;
   MOVE (NO-LC-MTAR) = (CTR-LG-MTAR);
   LET (LG-TRT-MTAR) = (CTR-LG-MTAR);
   IF (LG-TRT-MTAR) > 08 THEN GO TO FIN-PAGE-MTAR;
   MOVE (TAB-MTAR) = (WPAG-MTAR);
   GO TO MTAR-AFFICH;

  FIN-PAGE-MTAR:

   LET (CTR-PAGE-MTAR) = (CTR-PAGE-MTAR) + 1;
   IF (CTR-PAGE-MTAR) > 05 THEN
```

```
        DO
          UPDATE(FORM) MTAR,
          WINDOW=("TABLE PLEINE ; IMPOSSIBLE DE CREER !");
          MOVE (NO-LG-MTAR) = " ";
          LET (CTR-PAGE-MTAR) = (CTR-PAGE-MTAR) 1;
          LET (LG-TRT-MTAR) = 0;
          GO TO MTAR-AFFICH;
        DOEND;
      LET OFFSET(WPAG-MTAR) = OFFSET(WPAG-MTAR) + 0240;
      LET OFFSET(WPOS-MTAR) = 0;
      LET (PAGE-MAX-MTAR) = (CTR-PAGE-MTAR);
      MOVE (LIGNE-MTAR) = " ";
      MOVE (NO-LG-MTAR) = "01"
      LET (CTR-LG-MTAR) = 1;

      LET (LG-TRT-MTAR) = 1;
      GO TO MTAR-AFFICH;

REPRISE-SEQ-MTAR :

  LET OFFSET(WPOS-MTAR) = 0030*(CTR-LG-MTAR) 0030;
  MOVE (LIGNE-MTAR) = " ";
  MOVE (NO-LG-MTAR) = (CTR-LG-MTAR);
  MOVE (TAB-MTAR) = (WPAG-MTAR);
  LET (LG-TRT-MTAR) = (NO-LG-MTAR);
  GO TO MTAR-AFFICH;

FIN-MODIF-MTAR :

  MOVE (LIGNE-MTAR) = " ";
  MOVE (TAB-MTAR) = (WPAG-MTAR);
  LET (LG-TRT-MTAR) = 0;
  GO TO MTAR-AFFICH;


DDE-LG-MTAR :

  IF (CTR-PAGE-MTAR) = (PAGE-MAX-MTAR) THEN
    DO
      IF (NO-LG-MTAR) = (CTR-LG-MTAR) THEN
        DO
          MOVE (Z-ERR) = "NO-LG-MTAR";
          MOVE (MESS-ERR) = "IL N'Y A PAS DE LIGNE AU NUMERO DEMANDE";
          PERFORM AFFICH-ERREUR;
          GO TO MTAR-SAISIE;
        DOEND;
    DOEND
  ELSE
    DO
      IF (NO-LG-MTAR) > 08 THEN
        DO
          MOVE (Z-ERR) = "NO-LG-MTAR";
          MOVE (MESS-ERR) = "NUMERO DE LIGNE INFERIEUR OU EGAL A 08";
          PERFORM AFFICH-ERREUR;
          GO TO MTAR-SAISIE;
        DOEND;
    DOEND;
```

```
      MOVE (TAB-MTAR) = (WPAG-MTAR);
      LET (LG-TRT-MTAR) = 0;
      GO TO MTAR-AFFICH;
    DOEND;


MTAR-PAG-SUIV;

  IF (CTR-PAGE-MTAR) = (PAGE-MAX-MTAR) THEN
    DO
      PUT(FORM) MTAR
        WINDOW=("PAS DE PAGE SUIVANTE !"),
        LIST=(V-D-JOUR;TAB-MTAR);
      GO TO MTAR-SAISIE;
    DOEND
  ELSE
    DO
      LET (CTR-PAGE-MTAR) = (CTR-PAGE-MTAR) + 1;
      LET OFFSET(WPAG-MTAR) = OFFSET(WPAG-MTAR)+0240;
      MOVE (LIGNE-MTAR) = " ";
      IF (CTR-PAGE-MTAR) = (PAGE-MAX-MTAR) THEN
        DO
          MOVE (NO-LC-MTAR) = (CTR-LG-MTAR);
          LET (LG-TRT-MTAR) = (CTR-LG-MTAR);
          LET OFFSET(WPOS-MTAR) = 0030*(LG-TRT-MTAR)-0030;
        DOEND


          LET (LG-TRT-MTAR) = 0;
        DOEND;
      MOVE (TAB-MTAR) = (WPAG-MTAR);
      GO TO MTAR-AFFICH;
    DOEND;


MTAR-EFFAC:

  MOVE (LIGNE-MTAR)
  MOVE (NO-LC-MTAR) = (LG-TRT-MTAR);
  GO TO MTAR-AFFICH;


MTAR-COPIE;

  PROC HARDCOPY (VCOM(VEMU));
  GO TO MTAR-AFFICH;


MTAR-CONTROLE:

<<                                      >>
<<     TEST POUR DISTINGUER :           >>
<<         UNE SAISIE EN SEQUENCE       >>
<<         UNE DEMANDE DE LIGNE         >>
<<         UNE MODIFICATION DE LIGNE    >>
<<                                      >>

```

```
 920.5
 920.6
 940      ABANDON.
 940.1
 940.2      GO TO MCODE-INIT-ITEM.
 940.3
 940.4
 942      RETOUR-MENU.
 942.1
 942.2      SET(COMMAND)=EXIT;
 942.3
 942.4
 943      FIN.
 943.1
 943.2    END 030101
```

AUTOMATING SYSTEMS DEVELOPMENT WITH A DATA DICTIONARY

David C. Dummer
IMACS Systems Corporation
Los Angeles, California

This paper reviews the traditional role of a data dictionary and then considers its specific role and capabilities in automating application systems development. Delivery of the paper at the Edinburgh conference will concentrate on details of the types of computer aids that can be driven by Dictionary/3000; include practical experiences that the author has had with such aids; and make mention of the latest factual and speculative information on the future direction of Dictionary/3000.

Dictionary/3000 was introduced by Hewlett-Packard almost two years ago and there are now several other general purpose and specialized data dictionaries available for the HP3000 computer system from independent software suppliers.

Data dictionaries are in many ways the hot topic of the eighties much as data base management systems were in the seventies. Dictionaries are not new to the data processing profession, particularly in the mainframe environments, but they are undergoing great change as they evolve from a passive documentation role into being an active online component in computer system operations. The breadth of use and application of a data dictionary are still not well understood by the data processing and user communities. Even less well understood is the cost and effort Involved in setting up a data dictionary facility to ensure maximum benefits for both current and potential applications.

Data dictionaries first emerged as documentation and standardization tools and, indeed, these purposes still remain a major contribution to their use in data processing departments. This application of a dictionary can ensure that everyone within the department uses consistent names for objects, such as data fields and files, and understands the nature of an object when its name is used in verbal or written communications. This is not unlike a natural language dictionary that defines the meaning of a particular word (object) of the language and details its use in the context of other words.

Within the data processing environment we can identify further objects such as data entry screens, report layouts, data structures, data processing programs and computer system equipment. If each of these objects is given a unique name or identifier within its own type then use of the object should be clearly and consistently understood. This is becoming even more critical in distributed data base and processing situations where objects are often shared by many people in different locations.

For example, a data dictionary could define a data field, whose name is CUSTOMER-CODE, as an alpha-numeric string of ten characters which is used to uniquely identify a customer. The dictionary may provide still more information about the data field: synonym names that apply to the field in different areas of usage; physical location of the data field; the textual heading for the field when used in a report; validation rules for field values; who or what generates, changes or uses field values. This list is not meant to be all-inclusive but to simply highlight the fact that the dictionary can contain as little or as much information about an object as is required by the users.

Dictionaries are not new to many organizations. Most of them have at one time or another created manual forms to define data fields, data file record layouts and data processing record layouts. These dictionaries have normally been part of the documentation effort designed to allow additions and changes to existing files or programs to be readily implemented. However, the recent widespread computerization of data dictionaries has presented a tremendous opportunity in the scope of their usage. Considering that a data dictionary is simply an information system about a particular data processing environment, all of the tools available to support computer-based information management systems can therefore be used in the implementation of such a data dictionary. For instance, its content can be stored in a data base structure and use can be made of data base processing and reporting systems to effectively manage and utilize the contents. Unfortunately the computer industry has often been the last to benefit from its own technology and tools!

By making the dictionary resident on a computer system, its content is not only available to users but can also be accessed by programs and processes running on the computer. Such a dictionary data base becomes the hub of many computer systems and utilities that can aid the analysis, design, development, maintenance and control functions within the data processing department (Figure 1). The degree to which an organization evolves its data dictionary is generally a function of perceived value, the acceptance of centralized definitions and the implementation budget available. It should not be overlooked that a data dictionary will also require some level of maintenance and control on an ongoing basis.

Several organizations have created the position of data base administrator or information resource manager to address the need to manage the data resources of the company. As the requirement for more accurate and timely information for business decision-making continues to increase, the data base administration role becomes more focussed. The dictionary represents a fundamental tool by which the administrator can document and control the data processing environment under his or her charge. Since many groups within the data processing department will rely on the integrity of the dictionary contents, a data dictionary will typically employ some form of security that restricts its modification to the data base administrator. If an analyst or programmer produces new or modified systems then the resulting data dictionary entries should be made in a controlled manner through the data base administration function.

Many data dictionaries contain extensive directory features. This facility allows objects of different types to be linked in order to document ownership or usage. For example: a data field to a data file; a data field to a program; a data file (set) to a data base; a data file to a disk drive. The directory information not only supports system enhancement and maintenance efforts but can also allow programs to dynamically determine physical and logical relationships between objects requiring process. For example, a program may require a particular data field; information from the dictionary can then indicate which data file or files contain the field and, in a network environment, which computer(s) in the network should be accessed for the files. The directory organization of Dictionary/3000 is shown in Figure 2.

Some data dictionaries contain information about the data processing users and define which data files and programs a particular user may access. By placing this type of access security rule in the dictionary the data base administrator, or the equivalent function, can centrally control the data processing environment. Such a dictionary is of course complemented by the necessary programs, monitors and operating system interfaces that automatically effect and enforce the security measures.

Data dictionaries are clearly evolving towards a state where all or most of the definitions are removed from programs and placed in a central system data dictionary. These programs are then virtually independent of the physical structures within the data processing system and the structures can be modified, to react to configuration or usage changes, without affecting the programs. The transition from a passive to an active role has moved the data dictionary from a position of usefulness to one of necessity.

With this perspective of a computer-based active data dictionary and the definitional and directory structures available let us now examine its use by automated systems development aids.

The simplest computer aids are utilities that transform dictionary information into formats suitable for other processes to eliminate the need for any manual input effort. Examples of such utilities in the HP3000 environment are:

a) DICTDBC: an HP utility that uses an IMAGE data base definition in Dictionary/3000 to build a schema description file suitable as input to the DBSCHEMA processor. Generation of physical IMAGE data bases can therefore be completely automated from the design information in the dictionary.

b) IMACS*COBRA: an independent software vendor utility that builds COBOL copylib file records from data definitions in Dictionary/3000. COBOL program development can then benefit from the central definitions and control of the dictionary.

These utilities simply provide a 'bridge' to a process that was not originally designed to access the dictionary. One wonders, for example, whether a future release of the HP COBOL compiler might be capable of directly accessing the dictionary for most, if not all, of the information required for the DATA DIVISION. A direct link is however already used by the HP Transact compiler, a member of the HP RAPID family of productivity tools, which accesses the dictionary to resolve any undefined data items used in Transact program procedures (Figure 3). A programmer need therefore only define local computational items in a Transact program and can take advantage of central definition and redefinition in Dictionary/3000. The latter implies that no program changes or additions are required when data items, data bases or files are redesigned.

Processes that use dictionary information can also generate procedural logic based on defined relationships. Such processes further reduce the need for manual effort in building information systems. HP Inform uses the

Computer
Museum

Dictionary/3000 data base extensively to generate reports from requirements specified by a simple end-user menu dialogue (Figure 4). Inform first uses structures defined in the dictionary to present a set of tailored menus to the end-user from which he or she may explore the information available for reporting, whether stored in IMAGE data bases, KSAM files or MPE sequential files. This presents a fully logical view of the data to the end-user and is totally independent of the physical data structures. Once the end-user has indicated the data items required and any additional specifications (such as sorting, totals, selection rules, report format and computational fields), Inform accesses the dictionary to find out ·the file, or files, that contain the required data items. In the event of a data item appearing in more than one file, Inform uses an optimizing algorithm to minimize the number of files to be read and to favor files having some form of keyed access, such as an IMAGE data set. It should be noted that Dictionary/3000 provides for specifications that can override or influence the Inform algorithm. Inform also determines an optimal access strategy if more than one file needs to be read. From the dictionary information, indirect paths between files or data bases can also be determined, for example: the need to read one or more additional data sets to link two required data sets; or the use of a common data item not defined as a key, or match variable, as a link between two data sets or files. Inform finally builds the necessary procedural code to read the data file, or files, and to produce the specified report format.

Inform could of course have an independent set of tables or libraries to drive its algorithms and processes. Its use of a central dictionary minimizes the work of data processing staff in recording any system additions or changes and ensures that Inform has access to system information that should be the latest and most accurate available.

Inform is a good example of an automated system development tool for information reporting. However this is but only one part of general data processing - what about data entry, storage, manipulation and retrieval? Well, the same type of relationship structures used by Inform can also be employed to build procedural logic to support data entry and storage. In the HP3000 development environment, where VPLUS is a popular facility for block mode data entry, Dictionary/3000 can be used to define VPLUS form layouts and the relationship between form data fields and their appearance in storage files. Based on this information, algorithms can be designed to drive VPLUS functions and then the required IMAGE, KSAM or MPE operations for subsequent data storage or retrieval.

Variations of the developed procedural steps can be used to support other data processing functions such as editing and updating. Such procedural logic will often only support simple applications and, for instance, a data entry program specification may include complex rules for validation, error handling, table lookup, etc. However, many of these rules can be expressed in the form of definitions or tables and can therefore also be input to algorithms that will automatically generate the necessary procedural code.

There are many ways of providing the rules for automatic code generation and many language forms that the code can take. The rules can be provided by such vehicles as a very high level non-procedural system specification language, a set of definitions and rules in a data dictionary, the schema structure of an IMAGE data base or combination thereof. The code generated can be in such forms as COBOL, PASCAL, assembler statements or operators for a run-time transaction processor.

One example of the implementation of a dictionary-driven application generator is IMACS*PROGRAMMER. This product takes advantage of information in the Dictionary/3000 data base to produce Transact source code files. It can generate code to support the maintenance of IMAGE data bases, KSAM files and MPE sequential files with either a character mode or VPLUS user interface. Information from the dictionary can be complemented by additional specifications input by the person running the program generation process. The scope of Transact program generated varies from complete maintenance systems to sophisticated programs that perform only certain functions, such as data entry or reporting. The program produced may be a final solution or the program source code may be further tailored by a programmer to better suit the desired end-user interface or to perform specialized functions. IMACS*PROGRAMMER in particular supports a prototyping approach to application system development. The generation of each prototype can be completely, or largely, automated thus freeing the programmer to concentrate on the evolution process and derivation of final design.

So far we have considered utilities and processes that aid or automate application system implementation - what about analysis and design? We have seen how a data dictionary can document both physical and logical data processing structures. In the latter case this can clearly embrace the sort of data and process structures that evolve during the analysis and design steps of application development. Dictionary/3000 provides one structure, CATEGORY, that a user may employ to document logical and functional aspects of existing and planned systems. Even though the definition fields associated with CATEGORY are limited, a useful level of design aid can be implemented to document and assist in the analysis and design of procedures and data files. As standard requirements for such computer aids become more evident, Dictionary/3000 may well be expanded to meet their information needs.

There are several computer aids for analysis available in the mainframe environment. One example is STRADIS-DRAW from McDonnell-Douglas Automation. This product is based on the Gane and Sarson implementation of Data flow Diagrams that supports a system overview level and a design level for detailed specifications. A Tektronix color graphics terminal is used to enable an analyst to construct and manipulate the data flow diagrams. The diagrams and associated textual contents are maintained in a special file on the mainframe computer system.

In the HP3000 computer system environment IMACS*ANALYST is a computer aid that supports the well accepted Yourdon system analysis method (data-flow diagrams). The product uses a specialized dictionary data base to document the analytical components and relationships. However, as detail design work is accomplished the definitions of data items, files, bases and program procedures can be automatically loaded into Dictionary/3000. IMACS*ANALYST takes the automation level one step further by using Dictionary/3000 as a bridge between design and implementation.

It is not difficult to see the coming evolutionary steps for the dictionary and such computer aids. As more and more specifications and rules can be accommodated by the dictionary then more and more system development steps can be automated. Needless to say many challenges still exist in designing algorithms than can produce complex data processing systems.

An important first step has been made in the HP3000 computer system environment and you can expect to see many more computer aids from

Hewlett-Packard, independent software vendors and user group contributions that make use of Dictionary/3000.

Author Profile

David Dummer is president of IMACS Systems Corporation which develops and markets software products, education and consulting services through offices in the United States, Canada, the United Kingdom and the Netherlands. The company has specialized in productivity tools for the HP3000 computer system since 1977. Four of its software products were acquired by Hewlett-Packard in 1981, enhanced and now marketed as the RAPID product family of Dictionary, Transact, Report and Inform.

David has personally spent eighteen years in data processing most of which have been in the research, development and marketing of data base and dictionary driven systems for mainframe and mini-computer system environments. For several years as a data base administrator he gained considerable experience in shaping this corporate role and the associated development of data dictionary driven systems. He has lectured extensively in North America and Europe on technologies and implementation methodologies associated with the data base approach.

ROLE OF DICTIONARY FOR COMPUTER AIDS

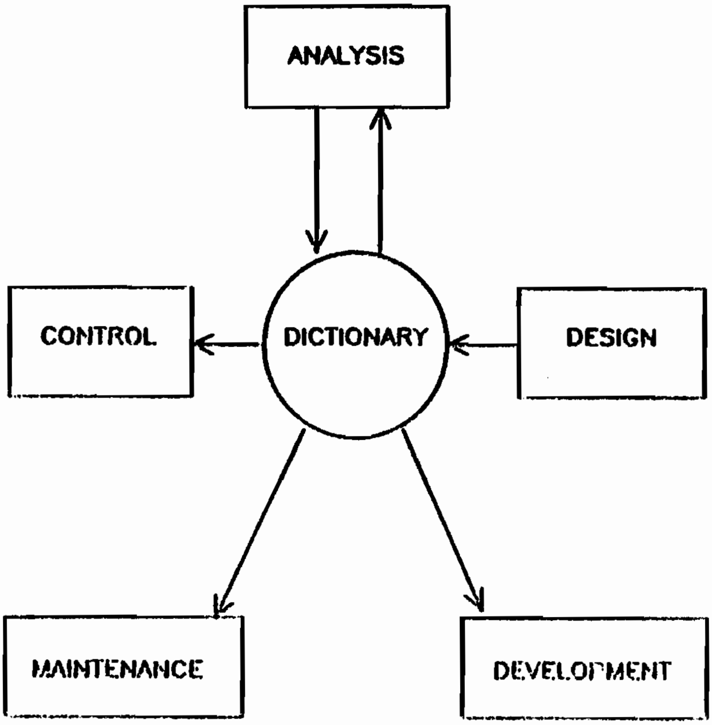

ANALYSIS
CONTROL
DICTIONARY
DESIGN
MAINTENANCE
DEVELOPMENT

FIGURE 1

DICTIONARY/3000: ORGANIZATION

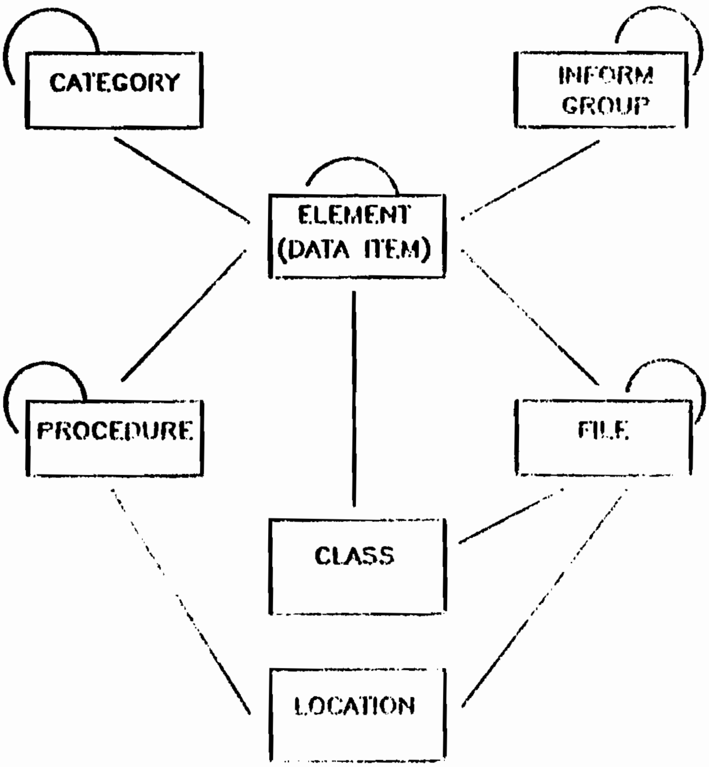

CATEGORY
INFORM GROUP
ELEMENT (DATA ITEM)
PROCEDURE
FILE
CLASS
LOCATION

FIGURE 2

TRANSACT/3000: ORGANIZATION

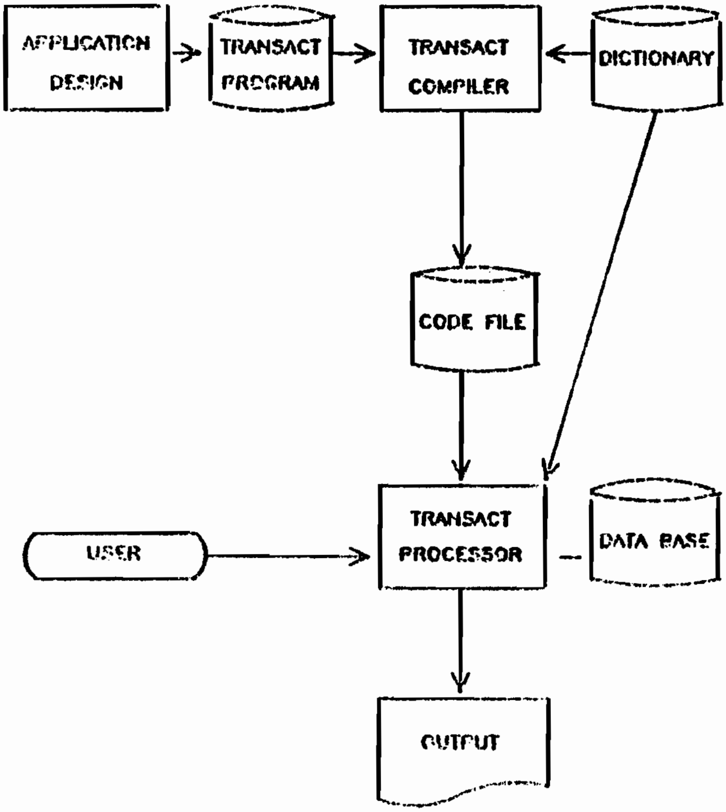

APPLICATION DESIGN
TRANSACT PROGRAM
TRANSACT COMPILER
DICTIONARY
CODE FILE
USER
TRANSACT PROCESSOR
DATA BASE
OUTPUT

FIGURE 3

INFORM/3000: REPORT GENERATION STEPS

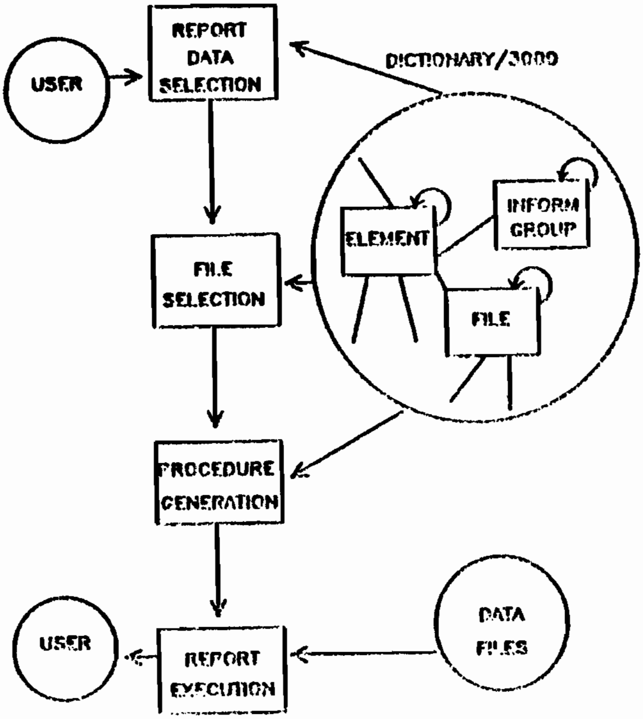

USER
REPORT DATA SELECTION
DICTIONARY/3000
FILE SELECTION
ELEMENT
INFORM GROUP
FILE
PROCEDURE GENERATION
USER
REPORT EXECUTION
DATA FILES

FIGURE 4

Data Dictionaries That Cost Nothing

by

Robert A. Evans

Freelance Analyst/Programmer

Sestil Ltd., 16, Lichfield Avenue,
Canterbury, Kent, CT1 3YA. 0227-58647.

To be presented at the

HP 3000 IUG International Meeting,

Edinburgh, October, 1983.

The object of this paper is to look at the process of system development, from database design through to program coding, in a slightly different light. The topics to be covered are:-

(a) How an IMAGE database is created, and how aspects of the creation process can be of particular use to us.

(b) The self-defining nature of IMAGE and VIEW files.

(c) Accessing and making use of this information, using standard software and simple own-written utility programs, and in program development.

(d) Conclusions to be drawn from the above.

Before starting, it is useful to set out some objectives, which are as follows:-

(a) Use standard Hewlett Packard bundled software, plus a compiler, wherever possible. Only use other development tools where they fill a need that cannot adequately be performed by standard software.

(b) Any utility programs that we write to make use of our data dictionaries should be simply and clearly written.

(c) Privileged mode is absolutely forbidden except where Hewlett Packard agree to support it. Failure to observe this guideline can leave us with a corrupt system and without support from Hewlett Packard. This topic is most important and has been fully covered by Eugene Volokh (1), although he draws a more lenient conclusion.

What are we looking for in this exercise? Some examples are shown below:-

(a) Database design aids - to ensure that our physical design is an efficient implementation of our logical design - e.g. selecting the right dataset capacity and block sizes.

(b) Program development aids - we can set up copy library members or record definitions automatically.

(c) Early warnings and indications of incompatibility - in a database, perhaps our sorted chains are becoming too large, our datasets are nearly full or are too full to be efficient. If, for example, we are using items of type I, they are not fully compatible with COBOL.

(d) Programming techniques that make use of the information available.

Every HP3000 installation has resources of data that are carefully looked after but only rarely used to the full. Data, for example, that would warn us that a particular database needs some attention. Data that describes the elements of each database and how the elements link together. Poor machine performance stems from many different causes, some of which are so elementary that they can be picked up and eliminated before calling upon the more complex performance monitoring tools. All this data can be accessed using standard I/O intrinsics from almost any programming language, at no extra cost. Only simple programming is involved.

Let us consider the stages we go through to set up and use an IMAGE database. We sketch out a design, having carried out our data analysis, and we usually have several attempts before we feel able to go any further. We describe the data and the way it links together. We define our datasets according to their type, and we work out the capacity we will need for each one. If we want to, we can do a dummy run through the schema processor so that we do not create a root file but get a listing in a familiar format that we can check and try to improve. We will probably make some small syntax errors, which we correct, and then when we are ready, we create the root file.

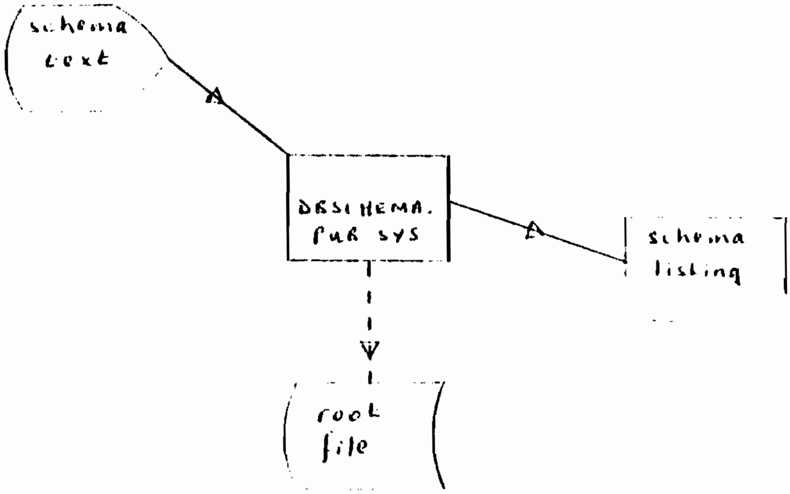


Fig 1: Schema Processor

Figure 1 shows this development stage. Notice the dotted line between the program and the root file. The latter is not created until (a) we ask for it and (b) the syntax is correct. THERE IS NO SUCH THING AS AN INVALID ROOT FILE. A schema input file can be invalid but not the root file.

All database designers should be aware of what will actually be set up on disk and the internal structure of an IMAGE database, as described in Section 10 of the manual (2). An important point that is not always realized is that the most suitable block size can be worked out individually for each dataset and set up using the CONTROL BLOCKMAX statement in front of each dataset definition (3). Spending a little time on this aspect of the physical database design can result in a more efficient database, which makes fewer demands on main storage at run time. Trial and error is the best way using CONTROL NOROOT and the Summary Table produced by the schema processor, which shows the amount of disk space needed by the root file.

Let us consider the root file. What is it, and what does it do?

(a) It is a machine-readable definition of our database and it holds details of all the items, sets, paths, capacities and security just as we set them up in the schema text file.

(b) It is the one and only reliable link between the files in our database and the I/O routines that Hewlett Packard provides to read it.

(c) It provides access to information about the current state of our database files. Some of this data is static (definitions); other is dynamic (record counts).

It therefore contains data that can be of much use to us. This point is taken up later.

To create the database, we run DBUTIL. By looking at the root file, DBUTIL finds out what is needed and creates the necessary files.

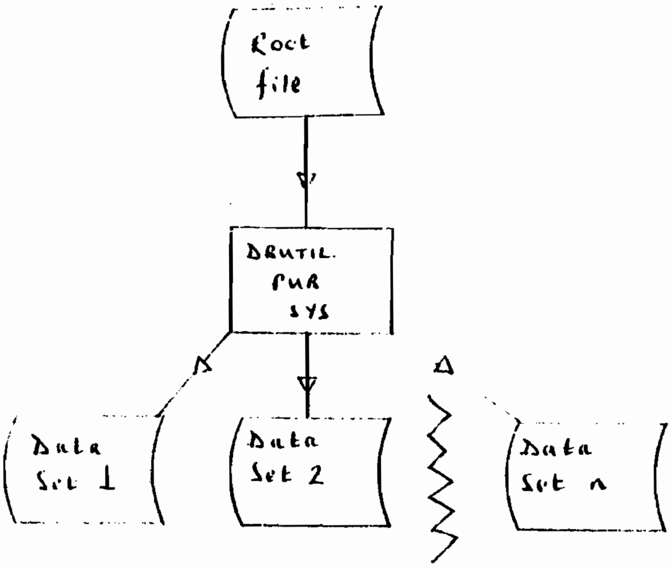

Root
file
DBUTIL.
PUB
SYS
Data
Set 1
Data
Set 2
Data
Set n

Fig. 2: Data Base Creation .

Figure 2, above, illustrates this process. By reference to the root file (not the schema text file), DBUTIL sets up the empty database. There is a further chance to enhance the physical database design here. Performance can be improved by suitable buffering (which is specified when running DBUTIL) and a guide to this technique is provided by Wendy Matheson in her paper "IMAGE/3000: Designing for Performance and Maintainability" (3).

Having got this far, with what by now should be an optimal physical database, we are now ready to put data into it by one means or another. At run time, using an IMAGE database, file access is performed as shown in Figure 3 below, with access to the data sets being controlled through the Data Base Control Block (DBCB) and with a User Control Block (ULCB) for each access path.

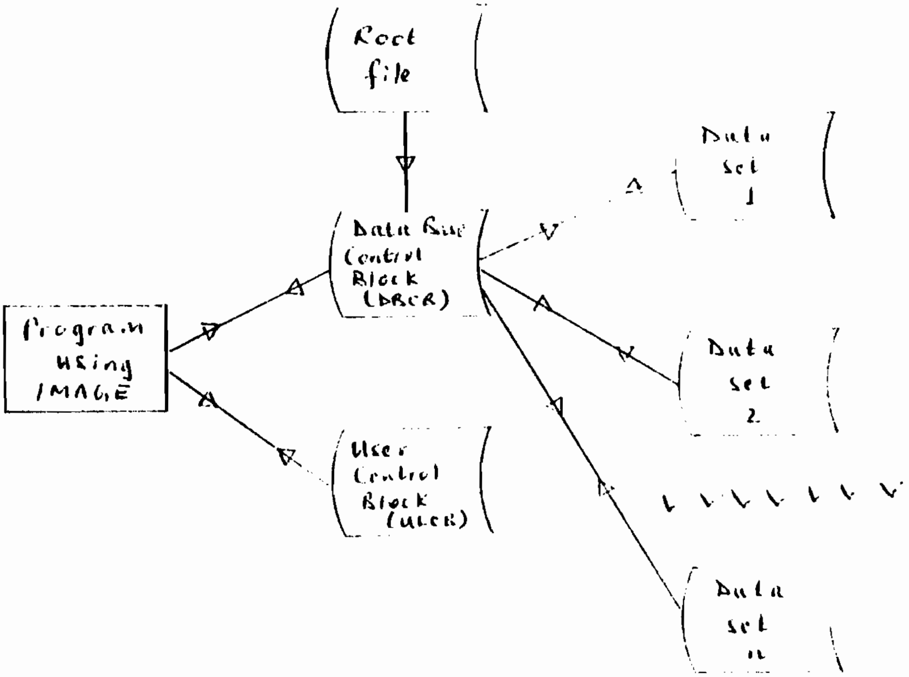

Root file
Data Base Control Block (DBCB)
Program using IMAGE
User Control Block (ULCB)
Data Set 1
Data Set 2
Data set n

Fig. 3: Data Base Access .

As Fig. 4 below shows, access to conventional files is simpler and data definitions are normally held in the program.

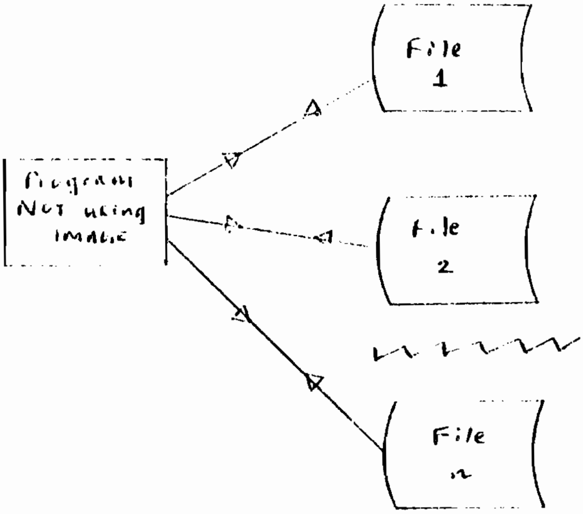


Fig. 4: Conventional File Access .

The big difference is that, when using IMAGE, file access also makes data definitions available whereas no such facility exists with conventional (or KSAM) files. This is how we are able to say: "Are there any INVOICE records with an INVOICE-DATE of 830601?" rather than: "Are there any INVOICE records with 830601 in positions 12 - 17?".

When accessing IMAGE data through a user-written program, not only are we presented with the record (or portions of record) that we ask for, but we are also given some very useful status information. The choice is ours whether or not to use it, but it is readily available in what is known as the Status Array.

The Status Array is "a ten-word array in which IMAGE returns Status information." (2) When the IMAGE procedure call works successfully, the array is set up with different information according to the intrinsic being called. The first word of the array is always returned from the procedure call with a value of zero if successful; non-zero if unsuccessful (with predetermined codes for each error condition).

A very simple read of a master dataset can be used to identify an impending loss of performance. Each master record's physical location in the dataset is determined by the type and value of the search item. For non-binary search items (type U, X, Z and P) IMAGE attempts to spread new records evenly around the dataset regardless of the value contained in the search item. Performance deteriorates when the dataset becomes around 80% full. For binary search items (type I, J, K, R) the search item VALUE determines the disk address where IMAGE will attempt to put the new record. Where a record already exists at this address, IMAGE has to find somewhere else nearby to put the new, Synonym, record. If an IMAGE program attempts to read this new record by search item value, it will first of all go to the 'old' one, realize that it is not the one required, and search through the synonym chain (records with the same calculated address) until it finds the one with the value it is looking for. As Wendy Matheson has shown, there need not be many records in the master data set for this to happen (3). Fortunately, we can detect this situation and do something about it.

Program 1: To check the number of synonyms in a master dataset (used when search item is type I, J, K, or R).

```
DBOPEN
DBGET  mode 2  (serial read)
Using the search item value thus obtained,
DBGET  mode 8  (calculated primary read)
Ignore if return code = 17 (record not a primary)

Words 5 - 6 then contain the number of records
in the synonym chain.
```

By maintaining a count of records read, and of synonym chains encountered, we can calculate some statistics. The program can be run periodically, and as a diagnostic when machine performance deteriorates. There is no "magic number" to watch out for, but trends can be established. The result should still be treated with caution. If the master detail set is always read serially, the program will deal with the records in the order it encounters them, regardless of whether or not they are synonyms, and no corrective action need be taken at this stage. If, however, the dataset is accessed frequently by search item value and Program I shows a trend of more and more synonyms, more and more I/O is needed to get hold of the right records. The solution is to examine the attributes of the search item, with a view to converting it to a non-binary type if possible, or perhaps adding a check digit (3).

Program 2: To look for overloaded sorted chains.

IMAGE offers the facility to sort detail records when adding into a chain. The manual suggests that these sorted chains be kept relatively short (2); Robert Green advocates a maximum of 10 (4). The following sequence of intrinsics, when carried out on a dataset containing a sorted chain, can be used to report on overloaded sorted chains:-

DBOPEN
DBGET mode 2 (serial read)
Using the (sorted) search item value just read
DBFIND
Words 5 - 6 of the Status Array contain the
count of the number of records in the sorted chain.
Report if not less than (say) 8.

Again, the results of such a program should be treated with caution. If records are rarely added to an offending dataset, or if they are added in batch mode at a convenient time, remedial action is less urgent. The solution is to consider whether the sort can be taken out of the chain.

Both of these programs can be written quickly in any language and in such a way as to take into account the nature of the processing in the individual installation. Privileged mode is not needed. These are just two examples of programs that make use of the Status Array which each program calling an IMAGE intrinsic has to set up. We can now go back a step to think again about the facilities offered by the IMAGE root file.

Every IMAGE root file is also a machine-readable data dictionary. When IMAGE datasets are accessed, the root file (or rather, a copy of it - the DBCB) is read at the same time to define the data and its structure. There is also an IMAGE intrinsic, sadly neglected, which can access these definitions programatically:-

DBINFO

Advantages of using IMAGE as a Data Dictionary.

1) An IMAGE ROOT File is always syntactically correct and up to date.

2) No effort is needed to maintain it. It sets itself up (DBSCHEMA.PUB.SYS).

3) There is no duplication, such as exists when a separate data dictionary is maintained.

4) Being up to date, there is no risk of incompatibility with the data it describes.

5) Before, during and after program development, where an existing database design is being modified, simply equate the root file to access the correct version.

6) Similar facilities are available in VIEW.

7) It costs nothing, and is delivered with every HP3000.

8) It is integrated with the data it describes.

9) It comes complete with a free, HP - supported means of interrogating it interactively (the FORM command within QUERY).

10) It is a mature, reliable product and there is plenty of expertise available. There is also a continual development and exchange of views on this product via the IUG.

The FORM command within QUERY.

This feature is fairly-well known. It has the facility to report on the usage of any, or all, of the data items, datasets, or paths within the database being accessed.

DBINFO.

DBINFO is an IMAGE intrinsic that provides information about any database and makes it available within a program. It should be pointed out that any such program should open the database with total read capability. With any capability less than this, it will not be presented with information for those items to which it does not have access. Some of the more commonly-used modes are shown below:-

Mode 203

"Identifies all datasets available in the database and the type of access allowed."(2).

The above mode of DBINFO returns a word array containing the set numbers for which the user has access. The numbers correspond to the order in which the datasets were originally defined in the schema.

Mode 202

"Describes a specific dataset."(2)
Having obtained the dataset numbers, we can now go in once for each dataset and obtain a lot more information:-

- dataset name
- dataset type (automatic or manual master, or detail)
- record size
- blocking factor
- capacity
- number of records present

Notice the combination of data definition and physical aspects.

We now turn to two practical uses of this information.

Program 3:

Access any database; find out each dataset name, capacity and number of records present. Produce a report showing this information and flag any dataset that is critically full.

For our definition of 'critically full', we can use that provided by David Greer - namely 80% for masters and 90% for details (5). The sequence of IMAGE intrinsics is:-

DBOPEN
DBINFO MODE 203 - what datasets are there?

DBINFO MODE 202 - capacities and record counts
Calculate, compare, print, and flag.

Once again, each shop needs to look at its own situation and requirements. For example, there could be a stand-alone detail dataset that always contains one and only one record, with a capacity also of one. This dataset will always be 100% full, and it might be worth hard-coding such a dataset out of the flagging routine.

Of particular value are flags that start to appear for master data sets whose search item is of a non-binary data type (U, X, Z and P). IMAGE attempts to spread these records evenly around the dataset and they start to get in each other's way (synonym chains begin to grow) when the record count approaches 80% of capacity.

Programming technique: looking ahead.

Consider a transaction that consists of adding one record to a master dataset and three records to a related detail dataset. If there is only room for one new master record and one new detail record, the transaction will fail, having added one record to each dataset, and leaving the database in a state of logical incompatibility, although perfectly valid as far as IMAGE is concerned. To avoid this situation, the program should first issue a DBINFO for each of the two datasets, and only continue if there is room for one more record in the master and three more in the detail dataset.

We can now continue to look at what is available in DBINFO:-

Mode 103 Defines all data items in the data base
and 104 " " " " " " data set
(and the type of access allowed).

A word array is returned, containing data item numbers. Having identified our datasets (modes 203 and 202) we are now well on the way to finding out how the database is made up.

Mode 102 Describes a specific data item.

This is the next step down and it tells us the na f a specific data item, along with its data type (I, J, K, R, U, X, Z, P), sub-item count and length.

Program 4: Create a COBOL copy library member or record definition for each dataset in any database.

```
Decide how to identify each set uniquely - prefixes?
DBOPEN
DBINFO  Mode 203 - what datasets are there?
        Mode 202 - define each dataset
Ignore automatic masters
        Mode 104 - what data items are there in
                   each dataset?
        Mode 102 - define each such data item
Convert data type to COBOL notation
e.g. J becomes PIC S9 (004) COMP
Deal with compound items
e.g. 30X2 becomes PIC X(002) OCCURS 30
Write to output file.
```

This program is easy to write, can be tailored to enforce an installation's programming standards, and can be used for any database once written.

Program 5: Report any data items that are not fully compatible with COBOL.

Data types I, K and R are not fully compatible with COBOL. If no processing is going to be done on any such fields in a COBOL program, there is no problem. Data type I is a particular problem because COBOL will happily deal with values within the range - 9999 to +9999 but not within the ranges - 32768 to - 10000 or 10000 to 32768. It would be useful to report any use of these data types for further investigation:-

```
DBOPEN
DBINFO  Mode 103 - what data items are there in
                   the database?
        Mode 102 - define each data item
Report data types I, K and R.
```

VIEW

Forms files also carry their data definitions around with them, and can be considered to be data dictionaries.

Again, as with IMAGE, we can access this information using standard software supplied by Hewlett Packard. Interactively, FORMSPEC'S LIST command is used to report one or more forms. Programmatically, three VIEW intrinsics can tell us everything we need to know about any forms file:-

VGETFILEINFO -	Returns FILE information - e.g. the FORMS it contains.
VGETFORMINFO -	Returns FORM information - e.g. the FIELDS it contains.
VGETFIELDINFO -	Returns FIELD information - i.e. the attributes of one or more fields.

We could write a program similar to program 4, which would use each of the three intrinsics in turn and set up a COBOL copy library member or record definition for each form. The method is slightly more complex, but information on each of the intrinsics is fully set out in the appropriate manual (6).

It is more than feasible to write a general purpose data-entry program that will quickly set up a dataset from a screen if certain naming conventions are followed. Bear in mind that the hyphen (-) character used in IMAGE corresponds to the underline (_) character used in VIEW and that fields in a VIEW form can be moved to their equivalents in an IMAGE database using the COBOL MOVE CORRESPONDING statement.

If we are using graphics, we can integrate DSG definitions with those of VIEW and IMAGE. Possibilities abound.

Conclusion

We have pursued several ideas through looking at the normal development process, but being a little more aware of what the software offers. All of the programs described are functionally simple, and can be quickly and easily written to suit each installation. Closer examination of the manuals and of technical papers will reveal further possibilities.

Only standard Hewlett Packard software has been considered. These products have been around for years and are reliable. Before buying any system development aids, we should look very closely at what can be achieved using IMAGE and VIEW. Furthermore, by adding to our knowledge of these products, we should be able to produce better systems.

By what might be termed good housekeeping, we can avoid some of the mysterious losses of performance and failures that occur from time to time.

I hope that I have generated a few useful ideas. I also hope that I have saved you some money.

References

1) VOLOKH, Eugene: Privileged Mode: Use and Abuse
VeSoft Consultants Interact Magazine September - October 1982.

2) Hewlett Packard Company: IMAGE Data Base Management System
Reference Manual.

3) MATHESON, Wendy: IMAGE/3000: Designing for Performance
Hewlett Packard Company and Maintainability IUG Montreal
April 1983.

4) GREEN, Robert M: Optimizing On-line programs.
Robelle Consulting Ltd. Technical Report 2nd edition.

5) GREER, David J: IMAGE/COBOL: Practical Guidelines
Robelle Consulting Ltd. IUG San Antonio Feb-March 1982.

6) Hewlett Packard Company: Data Entry and Forms
Management System
VPLUS/3000 Reference Manual.

A paper to be presented at:

HP3000 International Users Group,
International Conference,
Scotland.

2nd - 7th October, 1983

DIRECT SCREEN ADDRESSING
USING COBOL II

By: J. Birkhead,
L.C.P. Group Services Limited,
The Pensnett Estate,
Kingswinford,
West Midlands.
ENGLAND.

CONTENTS.

1 INTRODUCTION

2 LCP - WHAT IS IT?

3 DATA PROCESSING IN LCP

4 CURSOR ADDRESSING

5 DISADVANTAGES

6 ADVANTAGES

7 UPWARD COMPABILITY

8 PROGRAM SIZES

9 SUMMARY/CONCLUSION

10 BIBLIOGRAPHY

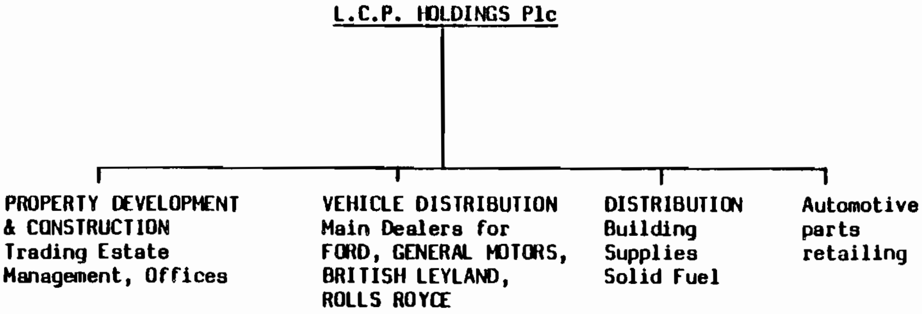
INTRODUCTION

This paper will highlight some of the reasons for and advantages/ disadvantages of using direct screen addressing with COBOL II at LCP. It should not be seen as an "anti-V/3000" paper, since V/3000 is an admirable product in the right environment, however, LCP felt that our installation was not such an environment for the reasons outlined later.

Descriptions of the method we use in formatting the screen will be given, although this example is by no means the only system that can be used.

LCP - WHAT IS IT?

LCP is a large group of diverse companies mainly based in the UK, but also in mainland Europe and U.S.A. The major activities of the group are outlined below:-


L.C.P. HOLDINGS Plc
PROPERTY DEVELOPMENT & CONSTRUCTION Trading Estate Management, Offices
VEHICLE DISTRIBUTION Main Dealers for FORD, GENERAL MOTORS, BRITISH LEYLAND, ROLLS ROYCE
DISTRIBUTION Building Supplies Solid Fuel
Automotive parts retailing

Most of the above companies data processing requirements are serviced by an in-house bureau based on a HP3000, with modems and multiplexors to regional depots remote from the central site.

Last year the group had a turnover of £277 million.

Continued...

DATA PROCESSING IN LCP

Five years ago D.P. was based on accounting machines. The information available from these machines was often late, inaccurate and inflexible. LCP leased an IBM System/32 and acquired a set of basic RPG ledger applications. Data was transferred over a period of 3 months from the accounting machines, these were eventually scrapped.

The IBM Syatem/32 was recognised as a stop-gap measure to build up the non-existant confidence in the Data Processing Department and computers in general. All data was keyed onto floppies and processed in batch through the 32. Eventually there existed 100 megabytes of backup storage on floppy - the system had 1 disk drive of 13.7 MB.

It was the eventual aim of the department to abandon central data-entry and to remove it to the point of transaction source in the user companies. In order for this to succeed, new applications were to be written that were as simple to use as possible.

The applications were also to be 'crash resistant', therefore it was decided to operate in foreground data entry/enquiry and background batch ledger updates. This method of working protects the major application files during a system crash. This has the effect of increasing user confidence.

Although in-house program development is not charged for, cost of the hardware to be used, i.e. VDUs and modems, was to be bourne by user companies. In order to reduce resistance to new applications, we recommended the use of the then newly available 2621As, instead of the more expensive 264X series. This of course meant the us
character mode.

Continued...

It should be remembered that at this time (1978-79) V3000 was relatively new, and HP S.E.s were rare in the UK.

The system/32 was replaced 18 months later by a HP3000 Series 33. A software house wrote a suite of on-line entry programs to interface the RPG background batch applications to the only method of data entry, the VDU. These programs did not use V/3000, but used their own screen handler in character mode.

With little experience of COBOL, two major applications were without a method of data entry. These were Payroll and Nominal Ledger. Programs were written using COBOL and V/3000.

Our own staff found V/3000 screens quite easy to use, however, when this method of entry was introduced to user staff, some resistance was encountered when they had used the character mode entry programs. Subsequently we have re-written the nominal ledger entry program.

After investigating direct cursor addressing and writing an application using it we decided that the method I will go on to describe is as good a method as any.

Continued...

Continued...

CURSOR ADDRESSING

Forms are placed on the screen using displays, with cursor addresses, followed by field descriptions. The use of WRITE to display operator prompts at the base of the screen and READ to accept user responses on the same line:-

ENTER CUSTOMER ACCOUNT NO.__

The effect "__" can also be achieved using ACCEPT and DISPLAY, however this is more difficult.

In the examples shown below the message field PROMPT-OUT is forty characters long. Longer fields can be created, but this then reduces the length of operator reply. Usually we use two PROMPT-OUT fields one of forty characters and one of 80 characters.

Essentially the screen is split into two sectors:-

Lines 0 - 19	Form Layout
Line 20	Blank
Line 21 - 22	Available for operator interaction

It should be noted that the 24 lines of the display are numbered 0 - 23 and the 80 columns are numbered 0 - 79.

Continued...

Continued...

A few of the more common escape sequences are described below:-

(? = Escape)

?H - Home cursor to top of memory

?J - Clear screen from current cursor position

?@ - Pause for one second

?&dD - Underline

?&dB - Inverse Video

?&a5r8C - Position cursor at absolute address row 5 (5r) and column 8 (8C)

When a cursor addressing program is run, the screen should be cleared (i.e. ?H?J). The blank form is then painted on the screen using display, as follows:-

```
.
.
.
CODE
.
.
.
.
DISPLAY "?H?J".
DISPLAY "?&a3r1C" "ACCOUNT NO.".
DISPLAY "?&a3r30C" "CUSTOMER NAME".
```

Continued...

Continued...

This will clear the screen, and write the words ACCOUNT NO. starting at row 3 column 1 and CUSTOMER NAME starting at row 3 column 30. The remaining form is built up in this way using lines 0 - 19 of the screen.

Below shows the basic statements that need to be entered.

Assign $STDINX is used to prevent E.O.D. when entering a COLON (:).

```
INPUT-OUTPUT SECTION.
      SELECT L-INP ASSIGN "$STDINX".
      SELECT L-OUT ASSIGN "$STDLIST".
             .
             .
             .
             .
             .
             .
DATA DIVISION.
FILE SECTION.

FD L-INP
      LABEL RECORDS MISSING.
01 REC-IN                                PIC X (10).

FD L-OUT
      LABEL RECORDS MISSING.
01 PROMPT-OUT                            PIC X (40).
             .
             .
             .
             .
             .
             .
PROCEDURE DIVISION.
A-OPEN SECTION.
      OPEN INPUT L-INP.
      OPEN OUTPUT L-OUT.
```

At this point the blank form is displayed as described above, prompt the operator and receive data as follows overleaf:-

Continued...

Continued...

```
ASK-AGAIN.
      DISPLAY "?&a20r0C ?J ?&a20r0C".
      MOVE "ENTER CUSTOMER ACCOUNT NO."
            TO PROMPT-OUT.
      WRITE PROMPT-OUT AFTER 0.
      MOVE SPACES TO L-INP.
      READ L-INP INTO ACCOUNT-NO
            AT END STOP-RUN.
```

This code will place the operators response (i.e. an account number) in the field ACCOUNT NO. This can then be used for validation.

If an error occurs, such as the account number does not exist, then follow the code below:-

```
IF CONDITION-CODE = 17 (IMAGE DBGET MODE 7)

      MOVE "ENTERED ACCOUNT NO. DOES NOT EXIST" TO PROMPT-OUT

      DISPLAY "?&a20r0C ?J ?&a20r0C"

      WRITE PROMPT-OUT AFTER 0

      DISPLAY "?&a22r0C ?@?@"

      GO TO ASK-AGAIN.
```

After the display of the error message the program will pause for 2 seconds and then the prompt for the account number is displayed again.

When the account number has been verified, it and the customer name can be displayed as below:-

```
DISPLAY "?&a3r15C" ACCOUNT-NO.

DISPLAY "?&a3r44C" CUSTOMER-NAME.
```

Continued...

Continued...

The entry of numeric values such as a unit price can be received from the screen using READ (into a PIC (X) field) then using a floating point sub-program be transferred to a PIC 9's field. However the COBOL II statement ACCEPT fieldname FREE can also be used. ACCEPT FREE allows the entry of numeric data, and automatically checks that the field is of the correct length, the correct number of decimal places and contains only numbers (or a decimal point or minus sign).

A field defined as:-

```
05 VALUE                 PIC 9(8) VALUE ZEROS.
```

Would take the operator input of 53 and place it in the field "VALUE" as 00000053. The inclusion of a decimal place or minus sign would institute an error. To allow the input of negatives the field should be defined as a signed numeric (i.e. S9(8)). To allow the input of decimal places define the field as S9(8)V99. An example of the input of a numeric field is shown below:-

Field defined as:-

Continued...

Continued...

```
05 VALUE           PIC S9(8)V99 VALUE ZEROS.
05 DIS-VALUE       PIC Z(7)9.99-.
.
.
.
.
.
CODE
.
.
.
GET-UNIT-PRICE.
      MOVE "ENTER THE UNIT PRICE OF THIS PRODUCT" TO PROMPT-OUT.
      DISPLAY "?&a20rOC ?J ?&a20rOC".
      WRITE PROMPT-OUT AFTER 0.
      ACCEPT VALUE FREE
            ON INPUT ERROR
               DISPLAY "?&a20rOC ?J ?&a20rOC"
               MOVE "INVALID ENTRY" TO PROMPT-OUT
               WRITE PROMPT-OUT AFTER 0
               DISPLAY "?&a22rOC?@?@"
               GO TO GET-UNIT-PRICE.
      MOVE VALUE TO DIS-VALUE.
      DISPLAY "?&a5rOC" DIS-VALUE.
```

(See COBOL II Reference Manual Section 11 pages 1 to 7)

When entering long text fields such as names and addresses, the displaying of a series of underscores can be used to indicate the remaining size of the field since these will disappear as the operator keys.

The program then continues to prompt for fields, validate, and display them as above.

The use of displays and writes can be reduced by using a message section. By moving a message number into a field and performing the message section, the use of cursor positioning can be kept in one place. By setting up program skeletons or copy libraries many of the standard prompts can be held in previously defined message sections or libraries.

Continued...

Continued...

By numbering fields during painting of the screen, at the end of entry the operator can be prompted to enter a field number should an amendment be required. Thus, the need for back tab to correct incorrect fields is removed.

DISADVANTAGES

There are obviously some disadvantages in using this method of data entry. Below are listed some of the major points.

a) When employing a HP experienced programmer we have to teach the programmer new techniques. This may encounter some resistance, however, in our experience the switch from V/3000 to direct cursor addressing can be achieved easily.

b) The absence of screen format files may mean some duplication of effort in creating the screen format again in another program. This may be overcome by using copy libraries.

c) Special code needs to be written to allow the refreshing of the screen after the break key is pressed or a power failure.

d) Any changes to the screen layout require a recompilation of the program.

Continued...

Continued...

ADVANTAGES

Below are listed the advantages as we see them.

a) A none HP programmer will learn escape sequences as easily as an interface to a screen driver. Consequently there is only the interface to IMAGE and KSAM to learn.

b) There is no software interface to have problems with. New versions of software should not effect escape sequences.

c) There is no significant difference in elapsed time when coding a cursor addressing program over using a screen handler.

d) Control of the screen is easier, the use of memory lock, display enhancements, and screen rolling allow the creation of complex screens relatively easily.

e) We have found that inexperienced user staff are easier to train using this method. Prompts have the effect of leading the operator through the screen.

f) If a field is in error, such as an invalid date, the operator is informed when return is pressed, not at the end of a screen.

Continued...

Continued...

g) There is never a large amount of data to transmit, therefore, the operator sees the system response more evenly since by the time the transaction is accepted all the data is in the machine and is error free.

h) Field prompts and error messages can be more descriptive, up to two or even three lines. This reduces the amount of field description narrative on the screen.

i) In a multi-user, multi-company environment there is no tendancy to proliferate screen format files which take disk space. Screen formats are merged with code, reducing directory entries.

j) We have found that the CPU is more evenly loaded since it is transmitting/receiving small amounts of information continually.

k) The use of character mode allows the use of system intrinsics to turn the echo off during password entry or timed reads which return to the program if no data is entered within a specified time limit, used to protect sensitive information.

Continued...

Continued...

UPWARD COMPATABILITY

One of the 3000 range of computers main selling points is that all software will remain upwardly compatible. This system of screen formatting is as upward compatible as any other since there are around 200,000 HP VDUs installed today. The moving of the cursor around the screen is controlled by hardware, not software so it "should" always work. V/3000 and any other screen handler must eventually produce the escape sequences that are created by the programmer. All this method does is to remove a software screen handler from between the COBOL program and the VDU. The new modular terminal family, however, has not yet been proven to work.

PROGRAM SIZES

In order to reduce the size of the non-reentrant data segments, all messages should be created and moved in the code segment, not in the data division. This has the effect of enbedding the screen handler in the code, so that only one copy is used in memory at once.

SUMMARY/CONCLUSION

In conclusion, we chose this method of screen formatting 4 years ago and have found no reason to change. There are obviously more character mode screen handlers generally available today, however, we find our own method quite acceptable.

Direct screen addressing can be as sophisticated or as simple as you want. We have found that with some practise, what may seem a laborious method of screen handling can be completed very quickly allowing the programmer to code, not worry about software interfaces.

Continued...

Continued...

If you are faced with the same problems we were then I would definately recommend this as an alternative to V/3000.

BIBLIOGRAPHY

Below are listed some references which may be of use in further reading.

1. 2645A Display Station Manual (HP) - Section II Display Memory Functions

2. Managing Data Entry with Non-standard terminals - 1980 SAN JOSE MEETING Section 7 - 17

3. Communicator Issue Number 23 (HP) - COBOL methods to control Data Prompts and Accepts to the same line of a Terminal.

SOME PRELIMINARY SUGGESTIONS FOR TRANSACT PROGRAMMING STANDARDS

by Tony Seymour and Lawrence McNamara

Air Call Computer Systems.

INTRODUCTION.

RAPID/3000 provides an integrated set of software tools which can dramatically improve programmer productivity in the development of new applications. One of these tools is the high-level programming language TRANSACT.

TRANSACT has been in use among HP users in Europe only for the last year or two. This has been a time for programmers to experiment with TRANSACT, while often finding themselves under considerable pressure to complete programs to tight deadlines due to high management expectations of "rapid" programming. As a result, there has been little or no time to consider questions of programming standards in TRANSACT.

It seems desirable to propose some initial standards for TRANSACT coding, on similar principles to standards already widely adopted for COBOL and other established programming languages. In this paper we will concentrate on the structure and layout of programs which handle VPLUS forms.

An example program, illustrating many points of this paper, is appended. The program is deliberately simple. It uses VPLUS forms to input, update, and delete names on the SIG RAPID mailing database.

STRUCTURE.

The overall design of the SIG program uses some of the facilities provided by TRANSACT for structured programming. The program comprises a number of hierarchically organized modules. At the top of the hierarchy is a control module (AA-CONTROL), which contains the only EXIT from the program. The control module says: keep repeating the menu until the user presses f8 (to exit), and it does this by means of the construct REPEAT PERFORM ... UNTIL ...

All other modules are performed routines, and each has a single entry point and a single exit (RETURN). The menu module (AB-MENU) calls one of 3 routines, BA-ADD, CA-CHANGE, or DA-DELETE, depending on the function key selected. The menu initiates an iteration of whichever routine is chosen, by means of the construct WHILE ... PERFORM ... The iteration is terminated when the user presses f7 (to return to the menu) or f8 to exit the program.

There is one other performed subroutine, AZ-FUNC-ERROR, which handles the invalid use of a function key anywhere in the program.

The use of unnecessary "GO TO" statements is avoided. The GO TO instruction is used mainly to loop back to the beginning of a paragraph or to proceed to the exit label of a module and thereby RETURN.

The demonstration program does not illustrate all the facilities provided by TRANSACT for structured programming. In retrieving information from IMAGE databases we prefer to use the FIND verb with the PERFORM=... option, and without the STATUS option. Performed modules called in this way should be handled in the same way as other modules.

LAYOUT.

The demonstration program illustrates many ways in which TRANSACT code can be easy to read, understand, and maintain. For example:

- Each module, and the handling of a new form within a module, begins on a new page in the listing.
- Each paragraph label occupies a line which is separated from adjacent code by lines of spaces.
- Space lines are used liberally elsewhere in the program to make it easier to read.
- There is a standard pattern of indentation. A new statement begins in column 3, while each option within the statement occupies a separate line, beginning in column 5. With conditional statements (IF... THEN... ELSE...) and the DO...DOEND construct there is further indentation.
- With the LIST verb, each item listed occupies a separate line, beginning in column 9.
- Comments are inserted where appropriate, although it is hoped that the program is self-documenting to a great extent.
- The name of a paragraph label is used both to describe the function of the routine and to aid in locating it within the program: e.g. BA-ADD.

VPLUS.

The demonstration program illustrates a method for handling VPLUS from TRANSACT. These are the main features.

- Variables are defined in the dictionary for
  (a) the name of the field to be enhanced (VENH)
  (b) the window message string (VMESS)
  (c) an integer to hold the number of the last function key used (VFKEY).

These 3 items are listed at the beginning of the program, and remain at the bottom of the list register. (At the start of each module, the list register is cleared down, and VFKEY becomes the current data item.)

- The function keys f7(return to menu) and f8(exit) are handled in such a way as to allow the user to return to the menu or to EXIT from anywhere in the program, while from the programmer's point of view this is handled in a structured manner. In every GET(FORM), PUT(FORM), AND UPDATE(FORM) statement the option FKEY=VFKEY is included, and f8= and f7= options are used to branch to the module-exit (RETURN). Control then passes back via the menu to the highest level of the program (AA-CONTROL) which checks the value of VFKEY.

- When handling a form, all the valid function keys (including f0 for ENTER) are specified as options. The invalid use of a function key can then be trapped without the use of an IF statement, simply by inserting a call to the subroutine AZ-FUNC-ERROR.

- The contents of the message window are controlled by including the option WINDOW=((VMESS)), in all statements which handle a form, and ensuring that the item VMESS is updated before such statements whenever necessary, e.g. by MOVE (VMESS) = " ";

- The typical handling of a form involves a maximum of 3 versions of GET(FORM).

  (1) GET(FORM) with the INIT option, to prompt for new input with the fields on the form initialized.
  (2) GET(FORM) with the CURRENT option, to re-prompt after the invalid use of a function key.
  (3) GET(FORM) with the CURRENT option, and with a field enhanced by WINDOW=((VENH),(VMESS)), after an error is trapped by program validation.

  The validation routine follows these 3 versions of the GET(FORM) statement, and is pointed to by the f0= option. If an error is encountered during validation, the items VENH (name of the field to be enhanced) and VMESS (message for the window) are set up and a GO TO statement causes the program to branch back to version (3) of the GET(FORM).

- The structuring of the program ensures that each module has a single RETURN in its exit paragraph. Just before this RETURN, any frozen form is cleared and old messages are removed from the field holding the window message.

This method for handling VPLUS from TRANSACT may not be the most efficient, nor the most compact in terms of lines of code. But in practice we have found that this style makes it possible to develop bug-free programs extremely quickly. Because of the standard manner of handling a form,it becomes possible to virtually generate much of a program with the text editor once the initial routines have been coded. A program written with this method is also very easy to read, which makes subsequent maintenance simpler - especially if someone other than the original author is asked to perform the amendments.

EFFICIENCY.

Coding standards for TRANSACT should also pay attention to the run-time efficiency of programs. Here the guidelines are already available in Appendix E of the second edition of the TRANSACT manual (published in December 1982).

- With large programs, segmentation should be considered. The organization of the demonstration program has this consideration in mind: the BA-ADD, CA-CHANGE, and DA-DELETE routines could later become segments 1, 2, and 3, leaving AA-CONTROL and AZ-FUNC-ERROR in segment 0. The structuring of of a program into modules makes eventual segmentation a simple matter. The sizes of segments may be assessed by compiling with the STAT option.

- In the VPLS= option of the SYSTEM statement, declare only those forms within the forms file which are required in the program. When the system is complete use FORMSPEC to compile to a fast forms file.

- The size of the list register should be kept to a minimum. Again, the structuring of the program helps because the SET(STACK) LIST(item-name) statement is used to clear down the list register at the beginning of each module.

- In the demonstration program the default TRANSACT data and and workspace registers are larger than required. Using test modes 3 and 102 the actual requirements were assessed, and the sizes of the registers tuned down by means of the DATA= and WORK= options of the SYSTEM statement.

- Where appropriate, temporary storage items defined in the program should use the OPT option, and in this case the program must be compiled with the OPTI option.

- A messages file or dataset may be used instead of program literals to hold strings for the message window. (This was not done in the demonstration program.)

CONCLUSION.

We hope that these preliminary suggestions for programming standards in TRANSACT will initiate some thought and discussion. If some agreement could be reached on coding standards, we believe that this would further improve productivity in coding, debugging, and maintenance work in TRANSACT, and also maximize the run-time efficiency of programs.

```
DICTIONARY/3000   DEFINITION OF DATABASE SIGRDB
---------------------------------------------------------------------


FILE                    TYPE: RESPONSIBILITY:
 SIGRDB                  BASE

FILE(ALIAS):     TYPE:     FILE(PRIMARY):       CAPACITY:
 A-NAMESET        AUTO      A-NAMESET            1001

        ELEMENT(ALIAS):                 PROPERTIES:          ELEMENT(PRIMARY):
         NAME                   *        X (30,0,30)          NAME


FILE(ALIAS):     TYPE:     FILE(PRIMARY):       CAPACITY:
 A-COMPANYSET     AUTO      A-COMPANYSET         1001

        ELEMENT(ALIAS):                 PROPERTIES:          ELEMENT(PRIMARY):
         COMPANY                *        X (30,0,30)          COMPANY


FILE(ALIAS):     TYPE:     FILE(PRIMARY):       CAPACITY:
 D-ADDRESS-DETL   DETL      D-ADDRESS-DETL       1002

        ELEMENT(ALIAS):                 PROPERTIES:          ELEMENT(PRIMARY):
         NAME                   *        X (30,0,30)          NAME
                                   CHAIN MASTER SET: A-NAMESET
         COMPANY                *        X (30,0,30)          COMPANY
                                   CHAIN MASTER SET: A-COMPANYSET
         ADD                            5X (30,0,30)          ADD
         TELNO                           X (20,0,20)          TELNO




DICTIONARY/3000   DEFINITION OF ELEMENT ADD (ADDRESS).
---------------------------------------------------------------------------


ELEMENT                 TYPE: SIZE: DEC: LENGTH: COUNT: RESPONSIBILITY:
 ADD                     X      30    0     30      5

LEVEL: ELEMENT:            POSITION:   PROPERTIES:
 (1) ADD1                    1            X (30,0,30)
 (1) ADD2                    31           X (30,0,30)
 (1) ADD3                    61           X (30,0,30)
 (1) ADD4                    91           X (30,0,30)
 (1) ADD5                    121          X (30,0,30)
```

DICTIONARY/3000 DEFINITION OF FORMS FILE VSIGR.

```
FILE                    TYPE: RESPONSIBILITY:
 VSIGR                   VPLS

FILE(ALIAS):       TYPE:     FILE(PRIMARY):
 VSIGMENU           FORM      VSIGMENU

 VSIGNAME           FORM      VSIGNAME

          ELEMENT(ALIAS):                PROPERTIES:            ELEMENT(PRIMARY):
           NAME                           X (30,0,30)            NAME

FILE(ALIAS):       TYPE:     FILE(PRIMARY):
 VSIGDETL           FORM      VSIGDETL

          ELEMENT(ALIAS):                PROPERTIES:            ELEMENT(PRIMARY):
           COMPANY                        X (30,0,30)            COMPANY
           ADD1                           X (30,0,30)            ADD1
           ADD2                           X (30,0,30)            ADD2
           ADD3                           X (30,0,30)            ADD3
           ADD4                           X (30,0,30)            ADD4
           ADD5                           X (30,0,30)            ADD5
           TELNO                          X (20,0,20)            TELNO
```

LISTING OF THE SIG DEMONSTRATION PROGRAM.

```
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:16 PM  COMPILED LIST


 COMPILING WITH OPTIONS: LIST,CODE,DICT,DEFN,ERRS,OPTI

                   <<*********************************************

                         Demonstration program.
                         Illustrates typical handling of VPLUS forms
                           with TRANSACT.

                         System inputs, updates, and deletes names
                           and addresses on the SIGRDB database
                           (HP SIG RAPID mailing list).

                         Written by Tony Seymour
                                 at Air Call Computer Systems

                                    July 1983.
                     *********************************************>>
```

```
   16.000
   17.000                 SYSTEM SIG,
   18.000   0000            BASE=SIGRDB("WRITER",1),
   19.000   0000            VPLS=VSIGR(VSIGMENU,
   20.000   0000                       VSIGNAME,
   21.000   0000                       VSIGDETL),
   22.000   0000            DATA=256,16,
   23.000   0000            WORK=256,32;
   24.000   0000
   25.000   0000
   26.000   0000            DEFINE(ITEM)
   27.000   0000
   28.000   0000                  SAVE-COMPANY     X(30),OPT;
   29.000   0000
   30.000   0000
   31.000   0000          AA-CONTROL:
   32.000   0000
   33.000   0000            LIST  VENH:
   34.000   0001                  VMESS,INIT:
   35.000   0003                  VFKEY;
   36.000   0004
   37.000   0004            REPEAT PERFORM AB-MENU
   38.000   0005 1                        UNTIL (VFKEY) = 8;
   39.000   0009
   40,000   0009            EXIT;
   41.000   0010          ((***********************************************))
   42.000   0010          !PAGE
1
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:17 PM  COMPILED LISTI


   43.000   0010          AB-MENU:
   44.000   0010
   45.000   0010            SET(STACK) LIST(VFKEY);
   46.000   0011
   47.000   0011          AB-01:
   48.000   0011
   49.000   0011            PUT(FORM) VSIGMENU,
   50.000   0011              WAIT=,
   51.000   0011              WINDOW=((VMESS)),
   52.000   0011              FKEY=VFKEY,
   53.000   0011              F8=AB-EXIT,
   54.000   0011              F1=AB-10,
   55.000   0011              F2=AB-20,
   56.000   0011              F3=AB-30;
   57.000   0025
   58.000   0025            PERFORM AZ-FUNC-ERROR;
   59.000   0027            GO TO AB-01;
   60.000   0029
   61.000   0029          AB-10:
   62.000   0029
   63.000   0029            WHILE (VFKEY) < 7
   64.000   0029              PERFORM BA-ADD;
   65.000   0036            GO TO AB-EXIT;
   66.000   0038
   67.000   0038          AB-20:
   68.000   0038
   69.000   0038            WHILE (VFKEY) < 7
   70.000   0038              PERFORM CA-CHANGE;
   71.000   0045            GO TO AB-EXIT;
```

```
0047
0047      AB-30:
0047
0047        WHILE (VFKEY) < 7
0047          PERFORM DA-DELETE;
0054        GO TO AB-EXIT;
0056
0056
0056      AB-EXIT:
0056
0056        RETURN;
0057      ((*************************************************))
0057
0057      AZ-FUNC-ERROR:
0057
0057        MOVE (VMESS) = " Invalid use of function key";
0059
0059        UPDATE(FORM) *,
0059          LIST=(),
0059          WINDOW=((VMESS));
0064
0064        RETURN;
0065      ((*************************************************))
0065      !PAGE
1
TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:17 PM  COMPILED LIST


0065      BA-ADD:
0065
0065      BA-10:
0065
0065      ((***********************************************
0065
0065          Handling of the form  VSIGNAME
0065
0065        *************************************************))
0065
0065        SET(STACK) LIST(VFKEY);
0066        MOVE (VMESS) = " ";
0068        LIST  NAME;
0069
0069      BA-20:
0069
0069        GET(FORM) VSIGNAME,
0069          INIT,
0069          LIST=(NAME),
0069          WINDOW=((VMESS)),
0069          FKEY=VFKEY,
0069          F8=BA-EXIT,
0069          F7=BA-EXIT,
0069          F0=BA-50;
0084
0084        PERFORM AZ-FUNC-ERROR;
0086
0086      BA-30:
0086
0086        GET(FORM) VSIGNAME,
0086          CURRENT,
0086          LIST=(NAME),
   128.000   0086          WINDOW=((VMESS)),
   129.000   0086          FKEY=VFKEY,
   130.000   0086          F8=BA-EXIT,
   131.000   0086          F7=BA-EXIT,
   132.000   0086          F0=BA-50;
   133.000   0101
   134.000   0101        PERFORM AZ-FUNC-ERROR;
   135.000   0103        GO TO BA-30;
   136.000   0105
   137.000   0105      BA-40:
   138.000   0105
   139.000   0105        GET(FORM) VSIGNAME,
   140.000   0105          CURRENT,
   141.000   0105          LIST=(NAME),
   142.000   0105          WINDOW=((VENH),(VMESS)),
   143.000   0105          FKEY=VFKEY,
   144.000   0105          F8=BA-EXIT,
   145.000   0105          F7=BA-EXIT,
   146.000   0105          F0=BA-50;
   147.000   0120
   148.000   0120        PERFORM AZ-FUNC-ERROR;
   149.000   0122        GO TO BA-40;
1
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:17 PM  COMPILED LISTI


   150.000   0124
   151.000   0124      BA-50:
   152.000   0124
   153.000   0124      ((  Validate that name does NOT already exist on file. ))
   154.000   0124
   155.000   0124        SET(KEY) LIST(NAME);
   156.000   0125        FIND(CHAIN) D-ADDRESS-DETL,
   157.000   0125          SINGLE,
   158.000   0125          LIST=();
   159.000   0128        IF STATUS = 1 THEN
   160.000   0128 1        DO
   161.000   0128 1           MOVE (VENH) = "NAME";
   162.000   0133 1           MOVE (VMESS) = " Name is already on file";
   163.000   0135 1           GO TO BA-40;
   164.000   0137 1        DOEND;
   165.000   0137      !PAGE
1
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:17 PM  COMPILED LISTI


   166.000   0137      BA-110:
   167.000   0137
   168.000   0137      ((**********************************************
   169.000   0137
   170.000   0137          Handling of the form  VSIGDETL
   171.000   0137
   172.000   0137        **********************************************))
   173.000   0137
   174.000   0137        MOVE (VMESS) = " ";
   175.000   0139        LIST  COMPANY:
   176.000   0140              ADD:
   177.000   0141              TELNO:
   178.000   0142
   179.000   0142      BA-120:
```

```
 180.000  0142
 181.000  0142           GET(FORM) VSIGDETL,
 182.000  0142             INIT,
 183.000  0142             LIST=(COMPANY,ADD1,ADD2,ADD3,ADD4,ADD5,TELNO),
 184.000  0142             WINDOW=((VMESS)),
 185.000  0142             FKEY=VFKEY,
 186.000  0142             F8=BA-EXIT,
 187.000  0142             F7=BA-EXIT,
 188.000  0142             F0=BA-150;
 189.000  0164
 190.000  0164           PERFORM AZ-FUNC-ERROR;
 191.000  0166
 192.000  0166         BA-130:
 193.000  0166
 194.000  0166           GET(FORM) VSIGDETL,
 195.000  0166             CURRENT,
 196.000  0166             LIST=(COMPANY,ADD1,ADD2,ADD3,ADD4,ADD5,TELNO),
 197.000  0166             WINDOW=((VMESS)),
 198.000  0166             FKEY=VFKEY,
 199.000  0166             F8=BA-EXIT,
 200.000  0166             F7=BA-EXIT,
 201.000  0166             F0=BA-150;
 202.000  0188
 203.000  0188           PERFORM AZ-FUNC-ERROR;
 204.000  0190           GO TO BA-130;
 205.000  0192
 206.000  0192
 207.000  0192         BA-150:
 208.000  0192
 209.000  0192           PUT D-ADDRESS-DETL,
 210.000  0192             LIST=(NAME : TELNO);
 211.000  0196
 212.000  0196
 213.000  0196         BA-EXIT:
 214.000  0196
 215.000  0196           SET(FORM) *,CLEAR;       (( clear a frozen form ))
 216.000  0198           MOVE (VMESS) = " ";      (( clear message window ))
 217.000  0200           RETURN;
 218.000  0201         ((*********************************************))
 219.000  0201         !PAGE
1
TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:17 PM  COMPILED LIST)

 220.000  0201         CA-CHANGE:
 221.000  0201
 222.000  0201         CA-10:
 223.000  0201
 224.000  0201         ((*********************************************
 225.000  0201
 226.000  0201             Handling of the form  VSIGNAME
 227.000  0201
 228.000  0201           *********************************************))
 229.000  0201
 230.000  0201           SET(STACK) LIST(VFKEY);
 231.000  0202           MOVE (VMESS) = " ";
 232.000  0204           LIST  NAME:
 233.000  0205                 COMPANY:
 234.000  0206                 ADD:
 235.000  0207                 TELNO:
  236.000   0208              SAVE-COMPANY;
  237.000   0209
  238.000   0209        CA-20:
  239.000   0209
  240.000   0209          GET(FORM) VSIGNAME,
  241.000   0209            INIT,
  242.000   0209            LIST=(NAME),
  243.000   0209            WINDOW=((VMESS)),
  244.000   0209            FKEY=VFKEY,
  245.000   0209            F8=CA-EXIT,
  246.000   0209            F7=CA-EXIT,
  247.000   0209            F0=CA-50;
  248.000   0224
  249.000   0224          PERFORM AZ-FUNC-ERROR;
  250.000   0226
  251.000   0226        CA-30:
 .252.000   0226
  253.000   0226          GET(FORM) VSIGNAME,
  254.000   0226            CURRENT,
  255.000   0226            LIST=(NAME),
  256.000   0226            WINDOW=((VMESS)),
  257.000   0226            FKEY=VFKEY,       .
  258.000   0226            F8=CA-EXIT,
  259.000   0226            F7=CA-EXIT,
  260.000   0226            F0=CA-50;
  261.000   0241
  262.000   0241          PERFORM AZ-FUNC-ERROR;
  263.000   0243          GO TO CA-30;
  264.000   0245
  265.000   0245        CA-40:
  266.000   0245
  267.000   0245          GET(FORM) VSIGNAME,
  268.000   0245            CURRENT,
  269.000   0245            LIST=(NAME),
  270.000   0245            WINDOW=((VENM),(VMESS)),
  271.000   0245            FKEY=VFKEY,
  272.000   0245            F8=CA-EXIT,
  273.000   0245            F7=CA-EXIT,
1
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:17 PM  COMPILED LISTI

  274.000   0245            F0=CA-50;
  275.000   0260
  276.000   0260          PERFORM AZ-FUNC-ERROR;
  277.000   0262          GO TO CA-40;
  278.000   0264
  279.000   0264        CA-50:
  280.000   0264
  281.000   0264        ((  Validate that name DOES already exist on file. ))
  282.000   0264
  283.000   0264          SET(KEY) LIST(NAME);
  284.000   0265          FIND(CHAIN) D-ADDRESS-DETL,
  285.000   0265            SINGLE,
  286.000   0265            LIST=(COMPANY : TELNO);
  287.000   0269          IF STATUS < 1 THEN
  288.000   0269 1          DO
  289.000   0269 1             MOVE (VENM) = "NAME";
  290.000   0274 1             MOVE (VMESS) = " Name is not on file";
  291.000   0276 1             GO TO CA-40;
```

```
   292.000   0278 1         DOEND;
   293.000   0278
   294.000   0278        MOVE (SAVE-COMPANY) = (COMPANY);
   295.000   0280       !PAGE
1
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:17 PM  COMPILED LIST)


   296.000   0280       CA-110:
   297.000   0280
   298.000   0280       ((*********************************************
   299.000   0280
   300.000   0280          Handling of the form  VSIGDETL
   301.000   0280
   302.000   0280         *********************************************))
   303.000   0280
   304.000   0280         MOVE (VMESS) = " ";
   305.000   0282         SET(FORM) VSIGDETL,
   306.000   0282           LIST=(COMPANY,ADD1,ADD2,ADD3,ADD4,ADD5,TELNO);
   307.000   0292
   308.000   0292       CA-120:
   309.000   0292
   310.000   0292         GET(FORM) VSIGDETL,
   311.000   0292           LIST=(COMPANY,ADD1,ADD2,ADD3,ADD4,ADD5,TELNO),
   312.000   0292           WINDOW=((VMESS)),
   313.000   0292           FKEY=VFKEY,
   314.000   0292           F8=CA-EXIT,
   315.000   0292           F7=CA-EXIT,
   316.000   0292           F0=CA-150;
   317.000   0314
   318.000   0314         PERFORM AZ-FUNC-ERROR;
   319.000   0316
   320.000   0316       CA-130:
   321.000   0316
   322.000   0316         GET(FORM) VSIGDETL,
   323.000   0316           CURRENT,
   324.000   0316           LIST=(COMPANY,ADD1,ADD2,ADD3,ADD4,ADD5,TELNO),
   325.000   0316           WINDOW=((VMESS)),
   326.000   0316           FKEY=VFKEY,
   327.000   0316           F8=CA-EXIT,
   328.000   0316           F7=CA-EXIT,
   329.000   0316           F0=CA-150;
   330.000   0338
   331.000   0338         PERFORM AZ-FUNC-ERROR;
   332.000   0340         GO TO CA-130;
   333.000   0342
   334.000   0342
   335.000   0342       CA-150:
   336.000   0342
   337.000   0342         IF (SAVE-COMPANY) = (COMPANY) THEN
   338.000   0342 1         UPDATE D-ADDRESS-DETL,
   339.000   0345 1           LIST=(ADD : TELNO)
   340.000   0345 1       ELSE
   341.000   0345 1            DO
   342.000   0351 1                SET(UPDATE) LIST(NAME);
   343.000   0353 1                SET(UPDATE) LIST(COMPANY);
   344.000   0355 1                SET(UPDATE) LIST(ADD);
   345.000   0357 1                SET(UPDATE) LIST(TELNO);
   346.000   0359 1                REPLACE(CURRENT) D-ADDRESS-DETL,
   347.000   0359 1                  LIST=(NAME : TELNO);
```

```
  348.000  0363 1              RESET(OPTION) UPDATE;
  349.000  0364 1           DOEND;
1
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:17 PM  COMPILED LIST]


  350.000  0364
  351.000  0364
  352.000  0364     CA-EXIT:
  353.000  0364
  354.000  0364       SET(FORM) *,CLEAR;      << clear a frozen form >>
  355.000  0366       MOVE (VMESS) = " ";     << clear message window >>
  356.000  0368       RETURN;
  357.000  0369     <<*********************************************>>
  358.000  0369     !PAGE
1
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:17 PM  COMPILED LIST]


  359.000  0369     DA-DELETE:
  360.000  0369
  361.000  0369     DA-10:
  362.000  0369
  363.000  0369     <<*********************************************
  364.000  0369
  365.000  0369         Handling of the form  VSIGNAME
  366.000  0369
  367.000  0369       *********************************************>>
  368.000  0369
  369.000  0369       SET(STACK) LIST(VFKEY);
  370.000  0370       MOVE (VMESS) = " ";
  371.000  0372       LIST  NAME:
  372.000  0373             COMPANY:
  373.000  0374             ADD:
  374.000  0375             TELNO;
  375.000  0376
  376.000  0376     DA-20:
  377.000  0376
  378.000  0376       GET(FORM) VSIGNAME,
  379.000  0376         INIT,
  380.000  0376         LIST=(NAME),
  381.000  0376         WINDOW=((VMESS)),
  382.000  0376         FKEY=VFKEY,
  383.000  0376         F8=DA-EXIT,
  384.000  0376         F7=DA-EXIT,
  385.000  0376         F0=DA-50;
  386.000  0391
  387.000  0391       PERFORM AZ-FUNC-ERROR;
  388.000  0393
  389.000  0393     DA-30:
  390.000  0393
  391.000  0393       GET(FORM) VSIGNAME,
  392.000  0393         CURRENT,
  393.000  0393         LIST=(NAME),
  394.000  0393         WINDOW=((VMESS)),
  395.000  0393         FKEY=VFKEY,
  396.000  0393         F8=DA-EXIT,
  397.000  0393         F7=DA-EXIT,
  398.000  0393         F0=DA-50;
  399.000  0408
```

```
     400.000   0408        PERFORM AZ-FUNC-ERROR;
     401.000   0410        GO TO DA-30;
     402.000   0412
     403.000   0412      DA-40:
     404.000   0412
     405.000   0412        GET(FORM) VSIGNAME,
     406.000   0412          CURRENT,
     407.000   0412          LIST=(NAME),
     408.000   0412          WINDOW=((VENH),(VMESS)),
     409.000   0412          FKEY=VFKEY,
     410.000   0412          F8=DA-EXIT,
     411.000   0412          F7=DA-EXIT,
     412.000   0412          F0=DA-50;
1
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:18 PM  COMPILED LISTI


     413.000   0427
     414.000   0427        PERFORM AZ-FUNC-ERROR;
     415.000   0429        GO TO DA-40;
     416.000   0431
     417.000   0431      DA-50:
     418.000   0431
     419.000   0431      ((  Validate that name DOES already exist on file. ))
     420.000   0431
     421.000   0431        SET(KEY) LIST(NAME);
     422.000   0432        FIND(CHAIN) D-ADDRESS-DETL,
     423.000   0432          SINGLE,
     424.000   0432          LIST=(COMPANY : TELNO);
     425.000   0436        IF STATUS < 1 THEN
     426.000   0436 1        DO
     427.000   0436 1          MOVE (VENH) = "NAME";
     428.000   0441 1          MOVE (VMESS) = " Name is not on file";
     429.000   0443 1          GO TO DA-40;
     430.000   0445 1        DOEND;
     431.000   0445      !PAGE
1
 TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:18 PM  COMPILED LISTI


     432.000   0445      DA-110:
     433.000   0445
     434.000   0445      ((*******************************************
     435.000   0445
     436.000   0445          Handling of the form  VSIGDETL
     437.000   0445
     438.000   0445        *******************************************))
     439.000   0445
     440.000   0445        MOVE (VMESS) = " Press f6 to delete name and address";
     441.000   0447
     442.000   0447      DA-120:
     443.000   0447
     444.000   0447        PUT(FORM) VSIGDETL,
     445.000   0447          WAIT=,
     446.000   0447          LIST=(COMPANY,ADD1,ADD2,ADD3,ADD4,ADD5,TELNO),
     447.000   0447          WINDOW=((VMESS)),
     448.000   0447          FKEY=VFKEY,
     449.000   0447          F8=DA-EXIT,
     450.000   0447          F7=DA-EXIT,
     451.000   0447          F6=DA-150;
```

```
  452.000   0469
  453.000   0469       MOVE (VMESS) =  " Invalid function key;"  +
  454.000   0472                       " use f6 to delete, or f7 or f8";
  455.000   0473
  456.000   0473     DA-130:
  457.000   0473
  458.000   0473       UPDATE(FORM) VSIGDETL,
  459.000   0473         WAIT=,
  460.000   0473         LIST=(),
  461.000   0473         WINDOW=((VMESS)),
  462.000   0473         FKEY=VFKEY,
  463.000   0473         F8=DA-EXIT,
  464.000   0473         F7=DA-EXIT,
  465.000   0473         F6=DA-150;
  466.000   0488
  467.000   0488       GO TO DA-130;
  468.000   0490
  469.000   0490
  470.000   0490     DA-150:
  471.000   0490
  472.000   0490       DELETE(CURRENT) D-ADDRESS-DETL,
  473.000   0490         LIST=(NAME : TELNO);
  474.000   0494
  475.000   0494
  476.000   0494     DA-EXIT:
  477.000   0494
  478.000   0494       SET(FORM) *,CLEAR;      (( clear a frozen form ))
  479.000   0496       MOVE (VMESS) = " ";     (( clear message window ))
  480.000   0498       RETURN;
  481.000   0499     ((*********************************************))

DATA ITEM DEFINITIONS:
  D  ADD                          5 X  (  30,   0,  30)
  D  ADD1                           X  (  30,   0,  30) = ADD(1)
1
TRANSACT/3000 COMPILER (A.00.03): FRI, JUL 29, 1983, 12:18 PM  COMPILED LIST


DATA ITEM DEFINITIONS:
  D  ADD2                           X  (  30,   0,  30) = ADD(31)
  D  ADD3                           X  (  30,   0,  30) = ADD(61)
  D  ADD4                           X  (  30,   0,  30) = ADD(91)
  D  ADD5                           X  (  30,   0,  30) = ADD(121)
  D  COMPANY                        X  (  30,   0,  30)
  D  NAME                           X  (  30,   0,  30)
  D  SAVE-COMPANY                   X  (  30,   0,  30)
  D  TELNO                          X  (  20,   0,  20)
  D  VENM                           X  (  16,   0,  16)
  D  VFKEY                          I  (   2,   0,   2)
  D  VMESS                          X  (  74,   0,  74)

CODE FILE STATUS: REPLACED


0 COMPILATION ERRORS
PROCESSOR TIME=00:00:46
  ELAPSED TIME=00:01:31
```

UNIX - AN INTRODUCTION

ABSTRACT

The paper explains why UNIX is of interest to the computing community at the moment, how it came into being, what its features are, and how it relates to CP/M, especially with respect to software portability; secondly, the ways in which UNIX can be used in a commercial environment are discussed including, by way of example, how Data Logic has used it as a software development tool and new product base.

Data Logic, a subsidiary of The Raytheon Company, supplies computer systems and services primarily in the U.K.

Original Author: M.J. Bailey, Managing Consultant.
Revised and presented by: John O'Leary, Design Consultant

Data Logic Limited,
Westway House,
320, Ruislip Road East,
Greenford,
Middlesex UB6 9BH,
England. (01-578-9111)

Data Logic

UNIX - AN INTRODUCTION

UNIX is often mentioned in the computer press these days, but how many readers know the reason for this, or even what UNIX is? If they know that UNIX is an Operating System, how many know what kind it is, and why it is so often compared with another Operating System, called CP/M?

Data Logic first became involved with the UNIX System (which is actually more than just an Operating System) about 3 years ago, and since then we have evaluated it, analysed it, and put it to practical commercial use, to the extent that we now have more than 50 designers and programmers who have worked with it. Currently we are undertaking both project work and studies involving the use of UNIX for systems and software development applications. So we are in a position to try to answer such frequently asked questions for those who wonder what this UNIX discussion is all about.

UNIX is in fact a Time Sharing System (TSS). It originally ran on a variety of minis, especially PDP-11's, but now runs on 16-bit and even 8-bit micros. The design is simple and the code compact, so the number of users who can be concurrently supported depends strictly on the hardware resources available, with few constraints from the Operating System.

It is ironic that, to some people at any rate, one of UNIX' rivals today is CP/M, which is a single-user-access, interactive, system, because UNIX was originally similar although it is now very much a multi-user-access system. In this sense UNIX was well ahead of its time when first developed, circa 1970, when systems were always either Batch (single-user, non-interactive) or TSS (multi-user, interactive). A TSS provides a much more friendly user-interface and quicker turn-around than a Batch system but unfortunately, in those days at any rate, TSS's only ran on large expensive mainframes. So UNIX was begun at the Bell Telephone Labs, in New Jersey, an Engineering/R&D environment, with the aim of creating a more productive programming environment than those available in the absence of a large TSS. The author of the first version (K. Thompson) found he had personal (i.e. single-user) access to a little-used DEC PDP-7 minicomputer, and he first developed the interactive aspects of a TSS on it, without the multi-user-access. UNIX was thus named for "uni-user MULTICS", after the TSS front-runner of that time, and possibly even now.

As the original program development facilities of UNIX became more popular at Bell, it soon evolved into a multi-user system, so as to make better use of the various computers available, and to this end it became portable as well. As a result UNIX is probably unique among Operating Systems, especially multi-user ones, in that it is over 90% written in a higher-level machine-independent language, namely, 'C' (a descendent of Cambridge University's BCPL). The early Assembler version was rewritten in 'C' in 1973 with only a (somewhat remarkable) 30% increase in code size which included adding multiprogramming to the system. The fact that it is mostly in 'C' gives it ease of modification as well as portability.

UNIX as we know it to-day has thus evolved as the system's requirements changed, and as the developers of the later versions (D.M. Ritchie and K. Thompson) evolved their ideas on the design of Operating Systems in small computers. So by 1974 the characteristics hoped for by its authors had become "simplicity, elegance and ease of use"; and because these hopes were essentially met (although "elegance" is in the eye of the beholder!), the UNIX user population mushroomed in the 1970's to 600 installations by 1978, and over 800 non-Bell installations by mid-1979.

This UNIX popularity was mostly in the Universities and R & D Labs, and was perhaps not unconnected with the fact that UNIX ran on the PDP-11 first and foremost, but it did lead to a tremendous growth in software "add-ons", developed by users and "in the public domain", in the line of compilers, text editors, debugging aids, document preparation tools (e.g. word processors, typesetting programs), etc.; plus a large number of less useful "recreation and novelty" programs. In addition, in view of the environments in which UNIX evolved it was gradually adapted for ease of use in experimental, real-time, situations, including early computer networks.

This evolution of the UNIX software in a small computer environment, with each increment of change developed on the UNIX system itself, has given UNIX the attributes which make it commercially attractive to-day, namely compactness, portability, reliability (or at least stability) and software abundance.

Essentially UNIX consists of a Basic Operating System, known as the "Kernel", plus over 100 "subsystems" which run as user processes controlled and scheduled by the Kernel. In a configuration which supports up to 50 simultaneously opened files, for example, and up to 64 concurrent processes, and has drivers for 6 different types of device, the Kernel occupies about 90Kbytes of memory on a 16-bit processor, and half of this is Buffer and Table Space (for files, devices and processes). This becomes about 110 Kbytes on a 32-bit processor.

The main features of UNIX are:

- A uniform system of file, device and inter-process I/O
- Ability to spawn autonomous processes from any process
- High degree of portability
- System Command Language is selectable on a per-user basis, and is not processed by the Kernel but by a user process
- Multi-level file naming and access control system, with dynamically growable files and mountable volumes.

The UNIX Kernel is in 3 parts, providing the following functions:

i) Process Management: process creation and (shared) program execution, swapping, process synchronisation, resource allocation, process scheduling.

ii) I/O System: secondary storage "Block" I/O with a Cache Buffer system, character device I/O, device drivers.

iii) File System: file naming, access control, disc space allocation, file sharing, "pipes" (i.e. inter-process communication via unnamed FIFO files).

On top of the Kernel software, the "subsystems" which make up the bulk of the system and run as user processes include:

- Login and Logout of Terminal User
- Output Spooler
- SHELL Command Language Interpreter
- Assemblers
- 'C' Object Code Binder
- Linking Loader
- Compilers for 'C', FORTRAN 77, BASIC, SNOBOL, APL, ALGOL 68, PASCAL
- Program verifier for 'C' programs

- Graphics packages
- Typesetting programs
- Word processors
- Syntax Analyser and Lexical Analyser
- Electronic Mail
- UNIX-to-UNIX File Copy
- 'MAKE' - Source and Object Code Maintenance System
- File Back-up and Archive

Commercial interest in UNIX began when the very cheap, very relisble, micro-chip processor arrived. Cheap, reliable, hardware is of no use without cheap, reliable, software to sell it, and UNIX had become a possible source for such software as far as some entrepreneurs are concerned, at least for 16-bit micros. CP/M, the single-user "Operating System" for 8-bit processors is another because it became the first Operating System developed purely for a micro to acquire a sizeable number of COBOL and BASIC commercial packages, initially for the personal computer market.

For this reason alone, although UNIX and CP/M are as different as chalk and cheese, they are often linked together in the probably irrelevant but apparently burning question as to which OS will become the "standard" for 16-bit micros. What they have in common is relative ease of software portability and a certain quantity of software but little else, and even their portability is achieved in quite different ways.

Most Operating Systems hitherto have been inextricably linked to a particular kind of hardware such that the only feasible way to port applications software from one hardware system to another, avoiding changes to the applications themselves, is to replicate the O.S. functions and file formats of the first system on the second, so that the porting or "conversion" task is typically:

1. Alter compilers' back-ends to generate second hardware code.
2. Rewrite O.S. in second hardware's Assembler language.
3. Debug the O.S.
4. Rewrite and debug the Run-time (High Level Language support) Library in second hardware's Assembler language.

5. Recompile and re-test all applications, utilities, compilers to be ported.

Needless to say this is very rarely done, although what is sometimes done is to emulate in microcode the first hardware's code on the second hardware, and possibly its O.S. interface too.

Both UNIX and CP/M simplify this porting/conversion task in different ways. The diagram shows two approaches to portability, and how High Level Language (HLL) programs progress from Source code to actual execution on the hardware.

UNIX uses Method A. CP/M, as far as its portable applications are concerned, uses Method B, and applications in this category are written in COBOL or BASIC.

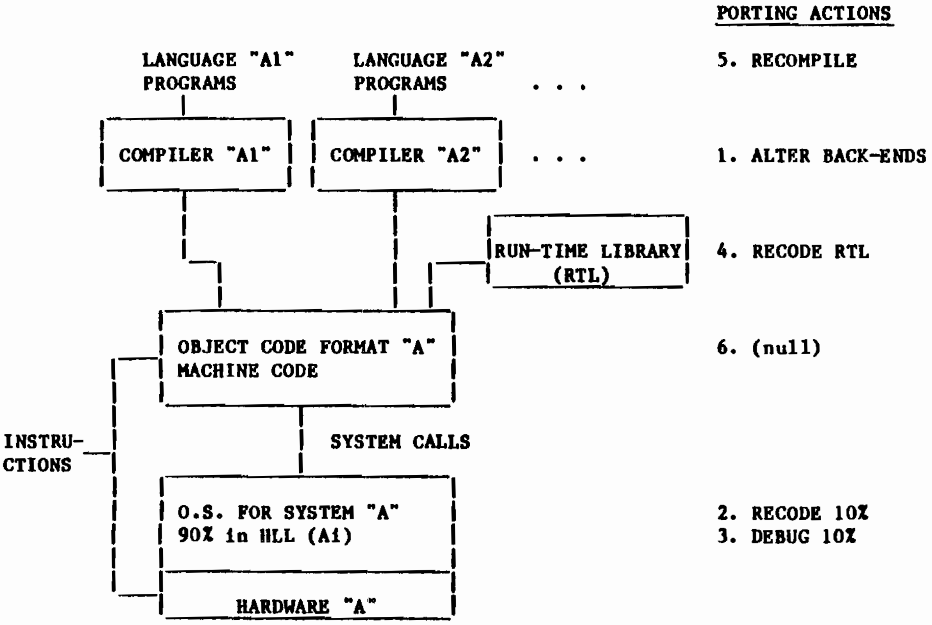
With UNIX, all five steps listed above are carried out (shown on the right in the diagram overleaf with the same step numbers). However the hardest steps (2 and 3) are greatly simplified because the O.S. is over 90% in HLL. Step 1 is also relatively easy because the 'C' compiler is constructed to generate symbolic assembler-like instructions, which are assembled into machine code by a separate program.

With CP/M (Method B) there is no recompilation, so no steps 1 and 5; but steps 2 and 3 must be carried out to the full, although CP/M is a relatively simple (and one-user) O.S.. Moveover with CP/M the COBOL and BASIC Run-Time Interpreters themselves must be rewritten as well as the Run-Time Library, therefore Step 4 is more difficult than in Method A.

Data Logic

TWO APPROACHES TO PORTABILITY

METHOD A



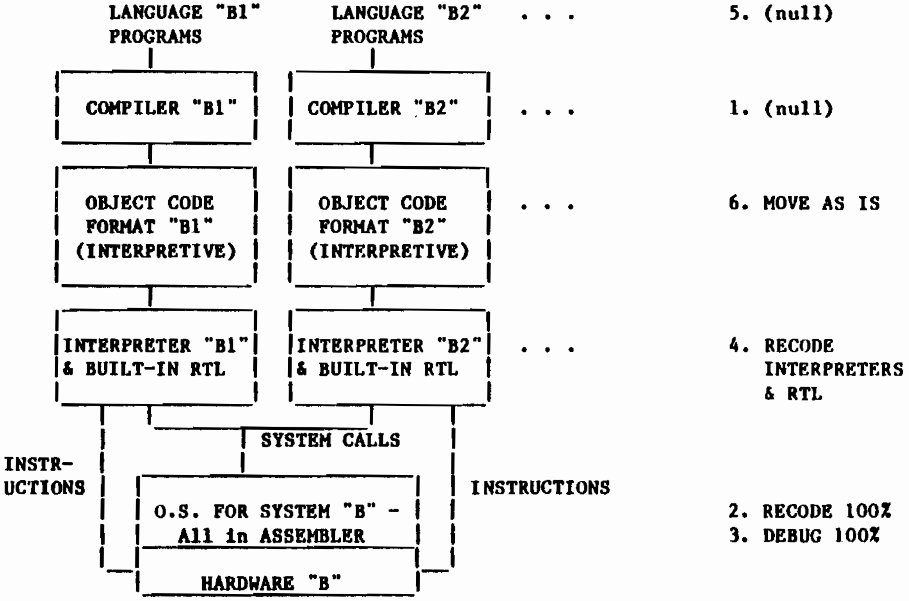
LANGUAGE "A1" PROGRAMS
LANGUAGE "A2" PROGRAMS
. . .
COMPILER "A1"
COMPILER "A2"
. . .
RUN-TIME LIBRARY (RTL)
OBJECT CODE FORMAT "A" MACHINE CODE
INSTRU-CTIONS
SYSTEM CALLS
O.S. FOR SYSTEM "A" 90% in HLL (A1)
HARDWARE "A"

PORTING ACTIONS

5. RECOMPILE
1. ALTER BACK-ENDS
4. RECODE RTL
6. (null)
2. RECODE 10%
3. DEBUG 10%

Data Logic

METHOD B



LANGUAGE "B1" PROGRAMS

LANGUAGE "B2" PROGRAMS

. . .

5. (null)

COMPILER "B1"

COMPILER "B2"

. . .

1. (null)

OBJECT CODE FORMAT "B1" (INTERPRETIVE)

OBJECT CODE FORMAT "B2" (INTERPRETIVE)

. . .

6. MOVE AS IS

INTERPRETER "B1" & BUILT-IN RTL

INTERPRETER "B2" & BUILT-IN RTL

. . .

4. RECODE INTERPRETERS & RTL

SYSTEM CALLS

INSTRUCTIONS

INSTRUCTIONS

O.S. FOR SYSTEM "B" - All in ASSEMBLER

HARDWARE "B"

2. RECODE 100%

3. DEBUG 100%

Two further points to notice with regard to the portability of UNIX and CP/M software are:-

i) that 'C' programs written to interface to the "C Portable Run-Time Library" only, i.e. containing no system calls, can also be run under IBM's OS or Honeywell's GCOS;

ii) application packages written in COBOL or BASIC to "run under CP/M" can in fact run under any system on which the Interpreters can run, provided that the system provides equivalent I/O facilities and file formats to those provided by CP/M - not the most difficult of tasks!

Therefore now that BASIC and COBOL Interpreters have been commercially developed for UNIX and/or its derivatives, the question as to whether UNIX or CP/M becomes a standard for applications on micros could soon become academic - to be superseded by "which BASIC and COBOL Interpretive Code Formats will become the standard for micro-based applications?"

Of how much use then is UNIX in the commercial world to-day? First it is a good multi-user development system that can run on mini and micro-based hardware. The programs developed using it need not necessarily run ultimately under UNIX, especially if they are written in 'C' and purely to the "C Portable Run-Time Library" Interface, or in interpreted COBOL or BASIC.

Secondly, UNIX can provide an instant product-base for multi-user interactive system software. This is particularly true if the system being developed is general-purpose and, if it is a multi-computer (distributed) system, if the configuration is a "star" (one master or cluster controller plus N satellites). In as much as there is a UNIX-to-UNIX File Store transfer capability, UNIX could also be theoretically adapted to local area network situations, i.e. linear or ring configurations, easier than some O.S.

Data Logic has been using UNIX for the last 2 years in the development of new software and hardware products, having evaluated UNIX both as a development system, and as a possible commercial product base.

We found that UNIX, Version 6 at any rate, had certain deficiencies to be overcome if used in a commercial product. The most serious of these were as follows (where "*" indicates a potential development system problem also):

*1. There was inadequate error detection and prevention (e.g. no file or record sharing locks, system crashes when the maximum number of open files exceeded, files could become corrupt after an I/O error).

*2. No recovery facilities, e.g. to do a "warm start", checkpoint a job, etc.

3. Access control to files and functions was not sufficient for some applications.

4. UNIX Shell Command Language was not user-friendly enough for application programmers or even some system programmers.

5. There were no Database Management Facilities, although there is nothing to prevent them being easily superimposed on the File I/O.

6. Each record accessed, in any user-imposed file organisation, required at least 2 disc accesses, at least 3 if the file is 70Kbytes, at least 4 if the file is 8.5Mbytes; this is worse if the file organisation involves intermediate records - e.g. ISAM.

*7. There was no per-user file space quota allocation/enforcement system and the system crashed when all file space was allocated.

*8. There was no Batch Job Scheduler - UNIX accepts as many background processes as it is given which eventually clogs the system.

9. Application Program Structuring problems could arise because there is no overlay facility and no program segment sharing between processes - only sharing of entire programs; the UNIX solution was to make each overlay or segment a process but this increases IPC overhead.

10. The IPC (Inter-process communication) facilities were not adequate. "Pipes" can only be used between a parent process and its child. There was no data segment sharing between processes.

11. All I/O appears synchronous to user processes; this fact, coupled with the inability to wait for the first of N events, created problems for "real-time" event handling processes such as Communications Servers; again the solution is to create N processes - 1 per event.

The principal remedies for these problems which we adopted were first for Data Logic to write a DBMS which sits on top of the UNIX file system and provides a relational database facility with 5 types of access path within such databases: SAM, ISAM, DAM, HRAM (Hashed-Random) and LIAM (Linked-Indexed, e.g. for WP).

The second remedy adopted was to use XENIX instead of UNIX, which helped us overcome such major problems as File System Integrity, inadequate IPC, and inadequate error detection/prevention. Other crucial problems, e.g. the lack of an overlay facility, were circumvented in various ways.

Using XENIX as a product-base probably saved us on the order of 25 skilled man-years when compared to starting from scratch. Product control software, the DBMS and the initial applications are all written in 'C' (only the most hardware-oriented pieces of special device drivers are in Assembler). Later, vertical market applications will be written in interpreted COBOL or BASIC which are supplied by outside vendors.

For development purposes we first used UNIX, then XENIX as its various enhancements became available, although the latter had little impact on the development system per se.

The main problems with UNIX as a development system are to do with ease of learning and, in our case (having a lot of software under development at one time) table overflow in the Kernel, 'C' compiler, and other utilities. The latter is fairly easily overcome by recompiling the offending part of UNIX, but learning to use UNIX effectively presented more difficulties because of the unprofessional quality of the UNIX user documentation.

This user documentation seems to be geared to Time Sharing System buffs who are not too long out of college. However once our systems specialists had learned, by experimentation sometimes, how UNIX should be used, it was relatively easy to pass this knowledge on to the application programmers, and it should be said straight away that we found UNIX to be a very powerful development system, especially in comparison to some commercial minicomputer (and even larger) systems that we have used.

The software MTBF was on the order of a week and the errors encountered were either of the aforementioned table overflow type, or very esoteric bugs.

Because of the portability of UNIX-based software discussed earlier we have been able to develop products first on a PDP-11, for later transfer to other machines whenever we wish to do so.

The faults in UNIX described here should not deter anyone from using it in the appropriate situations (as defined herein). Many of the problems listed, which Data Logic first recognised in 1980, are being rectified in various quarters, and there is certainly no comparable O.S. available for the price, especially one which is both so well tried over time, and so easily portable.

Data Logic

DEVELOPING LARGE INTEGRATED SYSTEMS USING RAPID/3000

A Presentation for:

The HP3000 International Users Group Conference Edinburgh 1983

By:

M.P. Ashdown
Shell UK Exploration and Production

DEVELOPING LARGE INTEGRATED SYSTEMS USING RAPID/3000

This paper describes the approach and methods adopted for a major development project - Project Management Information Systems, one of a number of concurrent development projects in Shell Expro.

SHELL EXPRO'S CONSTRUCTION PROJECTS

Before describing our own project I would first like to describe our user environment.

The development of oil and gas fields in the North Sea has necessitated the construction of large, complex platforms and production facilities in a hostile environment, with water depths of up to 500ft, waves of up to 80ft and wind speeds of up to 100mph.

Expro's philosophy for construction projects has evolved over nearly fifteen years of, sometimes bitter, experience and it is now common practice to set-up a dedicated 'project management team' for each large project.

The project teams have overall responsibility for the design, fabrication and installation of the platform, with most of the 'actual work' being contracted out. The projects have a duration of between three and six years and, at their peak, may have a management team of more than 200 staff. Costs for the larger projects can be well in excess of £500 million. Staff are assigned to projects from several functional departments, mainly Engineering, Materials, Quality Assurance and Project Services, to make-up a truly multi-functional and self sufficient team.

DEVELOPING LARGE INTEGRATED SYSTEMS USING RAPID/3000

To meet the current computing needs of project management teams and be prepared for a future, where more marginal fields requiring even more computing support are developed, Expro decided, in early 1981, to embark on a major exercise involving the development (60 manyears) of an integrated suite of 'Project Management Information Systems'. Here, I shall describe the application, the development environment and our experiences with the productivity tool RAPID.

THE PROJECT

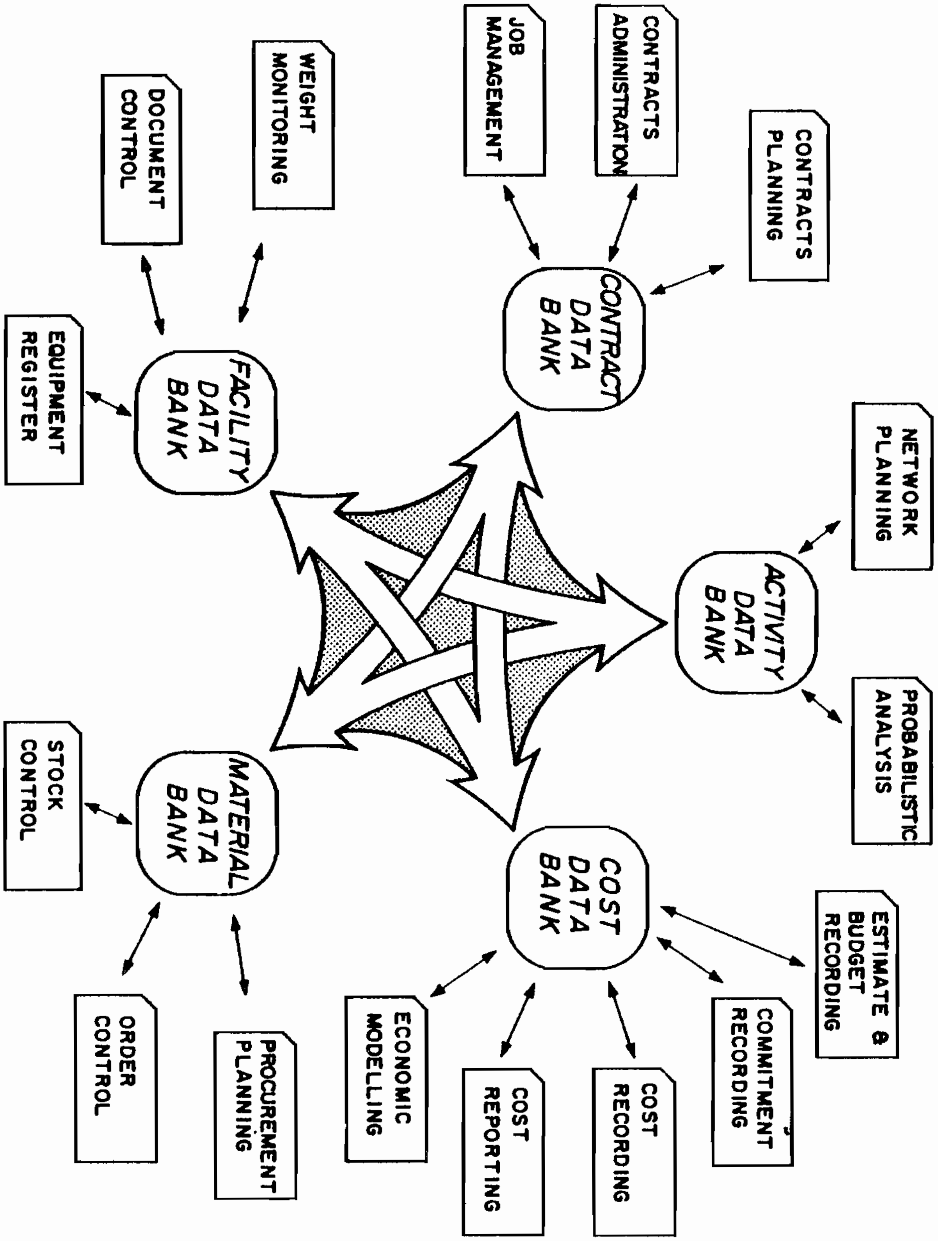
The use of computers by Expro's Project Management Teams increased rapidly in the late seventies with the introduction of dedicated minicomputers and associated 'do it yourself' software. Soon a variety of systems was developed to meet individual projects' urgent and specific needs but in the heat of battle, not enough consideration was given to relationships between the systems and their data.

It was decided to develop a comprehensive integrated 'Project Management Information System' for the benefit of all future projects and consisting of some sixteen modules (or systems). An outline description of these modules formed the framework for the 'PMIS' project.

Early analysis identified five main groups of data, related to the main business functions, comprising activity, cost, contract, material and facility data respectively. The sixteen computer systems to support the various business functions were grouped around the main 'data banks' as illustrated in figure 1. A closer look at the application of these systems within a construction project reveals that they are by no means independent; this being reflected by the interlocking arrows in the centre of the diagram.

Drawing No. 32204

PROJECT MANAGEMENT INFORMATION SYSTEMS


CONTRACTS PLANNING
CONTRACTS ADMINISTRATION
JOB MANAGEMENT
CONTRACT DATA BANK
WEIGHT MONITORING
DOCUMENT CONTROL
EQUIPMENT REGISTER
FACILITY DATA BANK
NETWORK PLANNING
ACTIVITY DATA BANK
PROBABILISTIC ANALYSIS
ESTIMATE & BUDGET RECORDING
COST DATA BANK
COMMITMENT RECORDING
COST RECORDING
COST REPORTING
ECONOMIC MODELLING
STOCK CONTROL
MATERIAL DATA BANK
PROCUREMENT PLANNING
ORDER CONTROL

DEVELOPING LARGE INTEGRATED SYSTEMS USING RAPID/3000

WHY THE HP3000

To achieve integration, the ability to share data between systems is essential and a simple software/hardware environment is, therefore, highly desirable.

Given the present state of communication facilities, we quickly found ourselves heading for a 'dedicated-mini' solution in order to satisfy the project team's need for interactive systems and to cater for a 'mobile' project team. (Construction project teams frequently operate from non-Shell offices and may even move during the project.) It was also clear that, certainly in large projects, the system could become heavily loaded adding 'growth path' to our list of requirements.

We selected the HP3000 on the following grounds:

- it is designed for an on-line transaction-oriented environment
- it has well integrated software and has good development tools
- it provides good communications facilities (needed for interfaces with corporate systems)
- there is an increasing range of commercially available packages which could meet some of our requirements, and
- there is considerable HP3000 experience in Shell UK.

We now have a 12 terminal-HP3000/44 (2Mb) with 520Mb (120+400) disc storage dedicated to the development and a similar system has been installed for the first projects' operational use. Our initial estimates indicate that large projects would have their own computer with up to 50 terminals and 800-1200Mb of disc storage.

SYSTEM CHARACTERISTICS

Because the systems are being developed to be run by project personnel, we have designed them to be as user-friendly as possible. A hierarchial menu structure has been used throughout, the exception being the G/C Cue Network Planning package (bought from a third party), but even this is accessed via a menu.

Conversational mode is normally used for retrieval purposes, and a 'fill-in-the-blanks' (VPLS) approach is widely used for data entry. In addition to a suite of standard reports for each system, a 'report-writer' (Inform & Report) facility is also provided.

Security provisions are based on password protected user names, thus allowing individuals to be granted (or denied) access to (sub) functions at any level of the menu hierarchy.

WHY RAPID/3000

At the start of PMIS in 1981 there were 6 dp staff in the development team, and we had just 3 years in which to provide 16 major systems, fully integrated and documented. We realised that we needed all the help we could find if we were going to meet our targets. So, while the first functional specifications were being written by our Users in the team, an evaluation was undertaken of productivity tools currently available on the Hewlett Packard 3000.

RAPID was the one chosen as a result of that evaluation on the following grounds:-

- Firstly, since RAPID/3000 is marketed by Hewlett Packard, this meant that not only could we buy our hardware and software tools from the same supplier, but also that if HP were to modify, say, the Operating System of the HP3000, we could be confident that RAPID would be made compatible, and its long term support and development were more certain.

- Secondly, RAPID is able to access all types of file structure on the HP3000, including multiple IMAGE databases and KSAM files. This was essential for PMIS, whose sub-systems included bought packages like G/C Cue. Whatever file structures they used, all of the systems had to be integrated.

- Thirdly, RAPID supports HP's packages for forms-handling and graphics (VPLUS and DSG); and, where necessary, COBOL, FORTRAN, SPL or PASCAL sub-programs can be used from within the TRANSACT main programs.

- Fourthly, initial demonstrations of the product and programs generated using it, showed that RAPID is 'User-friendly'; essential of course, for interactive systems such as PMIS.

So, in late 1981 we acquired RAPID/3000 and started to learn how to use it.

HOW WE USE RAPID/3000

DICTIONARY is central to our development; it contains the definitions of all our data items and their attributes and relationships enabling us to uniquely identify data items without any misunderstandings. These data items are first entered into the dictionary during analysis, this is as a by-product of documenting analysis within DICTIONARY.

Files and forms are defined for TRANSACT and REPORT programs and for INFORM reports. The programs are documented within DICTIONARY as are any predetermined reports.

We have written TRANSACT programs to extract information, in formats normally unavailable, from DICTIONARY so that we can show the relationships among elements, files, systems, etc... .

DICTIONARY also holds 'Help Text' which is available on-line to users whilst executing PMIS.

We use DICTIONARY and its utilities to create, maintain and enhance our databases.

INFORM groups are set up within DICTIONARY and both INFORM and REPORT are available to our users (although the use of REPORT is restricted due to the impact of compilations on system response).

To date 95% of our programs to date have been written in TRANSACT, the remainder in FORTRAN.

The approach to programming we developed was of a hierarchical modular design, the sixteen systems we required splitting nicely into five application areas. Each of the sixteen systems is to become a transact program (or system) belonging to one of five application area menus which in turn belong to the main menu in the structure. This methodology has given us very large TRANSACT programs. This set of menus and our security module became our master program which calls the systems available.

Within each of the systems the transactions are broken down into functional areas to allow sensible menu structuring. Normally each of the transactions is one segment; however in the case of some of the more complex transactions several segments are used. When this is the case logically complete parts of the transaction are put into their own segments.

This approach has allowed us to deliver systems independently and to easily test transactions during development.

A MATURING PRODUCT

When in late 1981 we acquired RAPID we were given a pre-release version and a training course by David Dummer. We experienced considerable problems for the next few months with bugs in the product stopping us or forcing us to use workrounds. However, our relationship with David and HP flourished and enabled us to have direct input into some of the early enhancements to the product.

When the A.00.01 version appeared we were starting to use more advanced techniques with the product, our staff having become familiar with its power. It was about this time that we began to see some of the shortcomings of the product. The documentation was very poor and as a result some misunderstandings arose; local HP support was almost non-existent; we constantly ran out of stack space.

We obtained advanced training and advice again from DCD and as a result our confidence in both ourselves and the product grew. We developed rules to stop known problems from occuring and workrounds to avoid other problems. We also wrote an 'in-house' manual to overcome some of the documentation shortcomings.

The A.00.03 release of transact helped stop problems of stack overflow by the use of the SWAP option on the CALL verb; documentation improved drastically which enabled us to discard some of our own in-house documentation and let us direct people to the manual instead.

In June 1983 we received A.00.05, a version of TRANSACT that we consider to be fairly stable. The last occurrences of integer overflows and bounds violations which had plagued it appear to have been removed and we are now confident we know how the product will perform/react under given circumstances. HP support is still poor (see later) but we only infrequently need help due to our own internal level of expertise. Our contact with the SE structure is more often for reporting faults.

Our use of RAPID over the last 2 years or so, has shown it to be an extremely effective tool for use in developing the PMIS systems. But it is not a 'perfect' product, and we believe that it may not be as well suited for other application areas.

LIMITATIONS OF RAPID

The main weaknesses that we have come across during its use, are, on the technical side first:

- a very limited mathematical capability and poor array (or table) handling. We have found the use of FORTRAN subroutines to be a satisfactory alternative to TRANSACT here.

- there is no equivalent to a Linkage Editor, so that if a change is made to code in one transaction, the entire system has to be recompiled. HP have been requested by ourselves and other clients to provide this Linkage Editor facility.

- TRANSACT currently 'locks' the whole database rather than just a single dataset from simultaneous updates by competing Users or proceses; this may be a disadvantage in some applications (HP say they will allow dataset locking).

- all our TRANSACT programs seem to require the maximum allowed data area at run-time, and therefore for the programs to run successfully, we have to use careful segmentation and good coding (see later).

Weaknesses of a more general nature, include:

- support for the product is still sub-standard. Very few HP Support Engineers have any experience with the product and 'experts' are virtually non-existent; the Phone In Consultancy Service also suffers from this lack of experience. Since it seems to be HP's policy to use SEs for customer training the level of tuition given to clients also suffers as the SEs are unsure of the material presented and often unable to answer questions.

- a significant learning curve. Just because TRANSACT is a high level, extremely concise programming language does not mean that it is necessarily simple to use. In fact, for coding of complex transactions which have to be precisely customised, TRANSACT can become as difficult to use as COBOL, however, it does retain the advantages of powerful verbs for database and screen handling. We have found that while a programmer can write basic programs after an introductory course and just a few week's use, it requires 3-6 months' experience and an advanced training course, before becoming fully proficient.

SOME SIMPLE TIPS

TRANSACT may be a free format language but in order to improve maintenance of programs coding standards should be developed e.g. for labelling conventions and indentation. (No one convention is necessarily better then an other but almost any is better than none.)

Ensure that programmers understand the implication of the verbs, their modifiers and options that they are using as significant performance gains can be achieved e.g. by using the SINGLE option with the FIND verb when only retrieving one record.

Avoid the use of the STATUS option as this removes a lot of power from a TRANSACT verb and importantly suppresses automatic error handling.

Encourage the use of test modes to debug programs (using 2 terminals, one equated to TRANDUMP, if you have enough VDUs to spare).

SOME ANSWERS TO THE STACK PROBLEM

Within the PMIS team this subject has become very important as we are now producing systems that have in the order of 25,000 words of p-code, for which the source is approximately 450-500 pages of standard print-out. This may not appear to be a problem unless you have produced a sizeable system using TRANSACT, but those that have, will probably have come across this one major problem of stack size.

The user data stack on the HP3000 is limited to 31,262 words (or less if your system configuration is less) and the TRANSACT processor has been PREPed to allow for this. The processor is effectively a command interpretor which operates on inputs from the user and upon the p-code generated by the compiler, thus the p-code resides in your user data stack. If you generate a large amount of p-code and do not segment your programs you will very quickly run out of stack.

Judicious use of the !SEGMENT instruction will solve most problems associated with stack size but does have other implications. Do not use only one !SEGMENT as the root segment (the first code in the source file before the !SEGMENT) always remains in the stack and this will be joined by the segment which is currently in use (or was just used if the current instruction is in the root segment); there will also be transfer vector tables for segment entry labels, so, if there are only two segments, there will be an overhead of the transfer vector table thus making the problem worse.

When segmenting keep the root segment as small as possible, because it is in the stack all the time, but put common routines in it if possible, as there is very little overhead associated with switching in and out of the root segment from the other segment in memory. All other segments should be kept as similarly sized as possible as the overlay areas are as large as the largest required by any one segment.

Remember, only those items defined in the root segment or defined in Dictionary and referenced in the root segment, are global to the whole program. Similarly, only the labels in the root segment are global and any labels outside this segment, that are referenced from a segment other than the one they are in, must be included in a DEFINE (ENTRY) instruction.

Global definition of items is important as internally each item is referenced by a number which is assigned as the compiler encounters them at compile time; thus the items in the root segment come first. The items referenced in all other segments but not occurring in the root segment are given numbers starting with the last item referenced in the root segment plus 1; however, this is done separately for each segment, thus the same number may be used for different items. This will never cause a problem if process control between segments is always passed through the root segment and if the only items in the list, each time control passes from the root segment, are items that have reference numbers generated from the root segment. The problems will arise if segment control is passed from one segment to another (and neither are the root segment). In this case, an item that is in the LIST, MATCH or UPDATE registers when the segment boundary is crossed, may have a reference number the same as a different item in the new segment, thus the processor would think a different item was in the register. To stop this occurring the processor checks that the references in the registers were not generated from the segment before it crosses the boundary, if any such reference is found an error message is generated. This checking takes time, so after thoroughly testing to ensure that the problem will not occur, use the OPTS compiler option for the operational system; this tells the processor that you are sure you have no local items in your registers when you change segments and thus not to check for you, thus eliminating the overhead.

The compiler offers two other chances to reduce the amount of stack required. Use OPT when defining items regardless of the segment, e.g.:

```
DEFINE(ITEM) REPLY X(1,,2), OPT;
```

This ensures that the compiler will not generate references for any item that is defined but not used within the segment or entire system (in the case of those defined in the root segment). Secondly, use the OPTI compiler option, in addition to OPT, so that all those items so defined will be stripped of their associated textual strings, such as their ASCII name and entry text, (assuming you do not use these strings in the system). The loss of ASCII names means that this should not be done on database items, as the name is required for the IMAGE call. Note that it is possible to define items that are already defined in Dicionary without altering their attributes but stripping these textual strings.

This can be very important when using the VPLS interface to remove strings associated with items only used for VPLS forms e.g.:

```
DEFINE(ITEM) FORMELEMENT,OPT;
```

[N.B.: Do not use this option on any item that you wish to highlight using variable field highlight i.e.

```
PUT (FORM) formname, WINDOW=((errorfield),(message)).]
```

If you do not require the heading text or entry text for database items in your programmes, but do require them for INFORM, it may be necessary to maintain two dictionaries. These strings cannot be stripped out at compilation time as the OPTI option would also take its ASCII name, but leaving these strings in may cause stack problems. Thus the only solution, currently, is to compile systems referencing a Dictionary that has not been given any of these strings, associated with the items, and running the systems and INFORM referring to one that does. Note that deleting a string from Dictionary is not the same as it never having had one.

Use of Test Mode 4 when running your systems can show you why and where your stack problems are occurring. This mode shows the following:

seg.address   instruction t_1 t_2 z s q dl where:

seg.address	is the segment number followed by the instruction location within the segment
instruction	is the p-code instruction at the given location
t_1	is the time taken to perform the instruction in millisecs (subtracting the test mode overhead) and if it is less than 1ms zero, is indicated and the fraction added to the next instruction

t_2	is the accumulated t_1 since the test mode was switched on
z	is the high water mark at the top of the stack
s	is the current top of stack
q	is the pointer that differentiates between global and local variables for the processor and its subroutines
dl	is the area in the stack taken up by MPE subsystems. Often VPLS.

This can show you how close you are to having stack problems as z+dl may never exceed 31,262 words.

Within the PMIS team we have a stack size calculation sheet (see below). These figures can be explained as follows:

1) PMIS menu overhead, this is caused by our systems being called from a master program using the SWAP option. This value is the summation of the remaining stack space after the call.

2) The stack required for the transact processor

3) The summation of the TRANSACT Processor Control Block and the Transact Outer Block

4) The DL required for VPLS

5) The bottom line of the compiler output with STAT option.

1, 2, 3 and 5 are explained further in Appendix E of the transact manual. The total we allow is 29.0 Kw which gives us a good confidence level that there is sufficient stack space left to allow for changes.

****

TRANSACT PROGRAM STACK SIZE CALCULATION SHEET

(1)	PMIS MENU OVERHEAD	1.5K	(fixed)
(2)	TRANSACT OVERHEAD	4.0K	(fixed - from HP)
(3)	CONTROL BLOCK OVERHEADS (TPCB & TOB)	1.0K	(fixed - from HP)
(4)	VPLUS OVERHEAD	....	(from fast forms file)
(5)	PROGRAM RUN-TIME STACK SIZE	....	(fom STAT listing)
	TOTAL :	....	

N.B. SYSTEM WILL NOT BE ACCEPTED BY THE SUPPORT TEAM IF THE TOTAL EXCEEDS 29.0K

****

THIS FORM MUST BE COMPLETED FOR EVERY SYSTEM

****

If you are using VPLS, the size of dl may be up to 8Kw which obviously limits room for other operations, such as, that other well known user of stack, the sort. A sort will require a minimum of 3Kw to run but will grab all of your stack available minus 1Kw to get it. You now have a problem if it cannot get enough stack as the dl pointer is so high. The only way to release this dl area is to close the forms file, unfortunately the CLOSE command will not do this. Solution: open a dummy forms file by doing a SET(FORM) on a form in the dummy file that has no fields.

Only one forms file can be open at one time, thus the first forms file is closed releasing the dl, the second is opened at a cost of less than 1Kw dl giving up to 7Kw for the sort. Do note that using fast forms files saves 800w and I/O.

All the preceding problems can be compounded by using the CALL command. Segmentation is limited to 64 segments and so, if large systems are to interface it may be necessary to use CALL. The problem with CALL is that it only shares the data register with the called system and builds all the other internal tables again in the data stack above the original ones, thus adding the calling system and called systems stack together. This problem can be overcome as of Version A.00.03, which has the SWAP option on the CALL command to allow the overwriting of some of the internal data when calling another system and these tables are rebuilt when crossing back. This obviously has an overhead but this is preferable to a system that will not run.

CONCLUSIONS

Naturally we have come to certain conclusions about the RAPID product, both from our own use of it and also from discussions with other users both within and outside Shell.

Firstly, we have indeed found RAPID to be an effective aid to productivity during the development of our PMIS systems.

However, we believe it could be less beneficial for 'batch' processing systems, rather than the on-line transaction-based systems for which it, and the HP3000 itself was designed. Also, it might be less effective for short-duration projects if their personnel had no previous knowledge of RAPID and who would therefore suffer from the learning curve.

In PMIS, we have made a rough estimate of 30-40% saving on programming, testing and documentation time, compared with if we had used a conventional programming language like COBOL.

The achievable gains in productivity from using TRANSACT are directly linked to the quality of staff employed for a particular task. High level languages do not mean that less skillful staff can be employed and expected to perform as well as the 'expert'. The areas in which skills are required change when using RAPID; it is possible to lose much of the potential for coding productivity by poor design, thus programmers and designers need to understand the concepts of RAPID in order to maximise their performance.

We have found RAPID to be a very powerful and flexible tool, limited only for mathematical and table-handling applications.

Despite having may be more than it is fair share of bugs in the initial months of its released in the UK, it is now a stable and reliable product, which it appears HP have every intention of supporting and developing further.

RAPID's 'user-friendliness' and performance in development and 'live' environments, have been found generaly to be very good, and compare favourably with COBOL.

Lastly, it should be mentioned that RAPID goes beyond the normal realm of a development tool by providing the Support Team, who have to maintain the Systems, both with good documentation of data and program structures, and also with utilities to maintain and audit the databases and their contents. The source code that has been developed has so far been easily maintained and enhanced leading us to hope for productivity gains from our support team. It is however too early in the life of our operational systems to be able to quantify any such improvements.

PROGRAMMING
FOR
PERFORMANCE

This paper represents my personal concepts of programming and performance as best I can express them at this time. Nothing I have either included or excluded is meant to represent the official posture of the Hewlett-Packard Company.

Hewlett-Packard allows, expects, and encourages individual initiative. The existence of this paper exemplifies that philosophy in action. Hewlett-Packard has completely supported my efforts. I have enjoyed total freedom in my choices of subject, content and format.

This paper originated from the Baltimore, Maryland sales and service district office of the Eastern Sales Region. It was physically prepared on equipment in the Technical Center in Rockville, Maryland.

The graphics were created on a 2647A terminal using Interactive Formating System software. Textual data, sample programs excepted, were prepared using HPSLATE. Final printing was done on the 2680 Laser Printer. A COBOL program using IFS intrinsics generated input to the printer.

I have had much help and cooperation in preparing this paper. Thanks to all with special appreciation to Ruth, my wife, for her patience, support and unselfish sacrifice of many evenings and weekends during its preparation.

SECTION 1 GENERAL INTRODUCTION

- -- SEQUENTIAL PROCESSING LIMITED BY DISC
- -- FORTRAN OUTPERFORMS COBOL
- -- V/3000 TOO INEFFICIENT
- -- IMAGE IS TOO SLOW
- -- HP3000 INEFFECTIVE BATCH PROCESSOR
- -- MOON IS MADE OF GREEN CHEESE

Figure 1

TRADITION (Figure 1)

-- SEQUENTIAL PROCESSING IS LIMITED BY DISC ACCESS

This is one of those generally accepted truths that all of us have been taught from the beginning of our careers. In most cases, on most machines, this is probably a valid generality. Because HP has concentrated on transaction processing, certain defaults built into the file system often make this assumption invalid when evaluating batch processing performance.

-- FORTRAN OUTPERFORMS COBOL.

In typical business applications, this is generally untrue. The FORTRAN compiler is admittedly superior in handling numeric data if we limit the definition of numeric data to binary and real data formats. The COBOL compiler, however, does a much better job with ASCII numeric data and is quite effective when dealing with files, records, and ASCII character fields. The net effect is that COBOL usually outperforms FORTRAN in the average commercial application.

-- V/3000 IS TOO INEFFICIENT.

Considering the powerful, highly generalized capabilities provided within V/3000, this statement is not acceptable. The original V/3000 did have design characteristics, particularly the use of KSAM and the prohibitive form file recompilation techniques, that left a bad taste in our mouths. The new V/3000 has corrected these shortcomings. In any particular application, a good programmer could probably outperform V/3000; even so, V/3000 is a highly efficient subsystem that deserves the chance to earn our confidence.

-- IMAGE IS TOO SLOW.

IMAGE is not at all slow but it is easily abused. IMAGE is an excellent example of a network data base and as such is inherently very rapid when used for record retrieval. Unfortunately, the price of rapid random retrieval is relatively slow structure maintenance, particularly when the structure becomes complex. Add to this the overhead for sophisticated internal security, multiuser update access, and extensive chain sequencing and you have a heavily burdened environment. In effect, the demands of our applications lead to slow performance; IMAGE itself is not inherently slow.

-- HP 3000 IS INEFFECTIVE IN BATCH PROCESSING.

The HP 3000 running MPE is admittedly biased towards transaction processing. The defaults built into MPE have not been chosen to maximize batch processing; the programmer who attacks a batch application without taking this into consideration may be displeased with the results. An informed programmer will know which override options to invoke in order to bring out the batch processing strengths of the machine. When handled properly, the HP 3000 is capable of surprising batch performance.

-- THE MOON IS MADE OF GREEN CHEESE.

I finally stopped believing this in 1969.

PROGRAMMING FOR PERFORMANCE

-- Programmer is problem solver	-- Measured by your MANAGEMENT!
-- Programming is problem solving	-- Compromise between elements
-- Entire problem is fair game	-- Doing job -on schedule -within budget
-- Entire solution is fair game	-- Response (TP & B)
	-- Throughput (TP & B)

Figure 2

PROGRAMMING AND PERFORMANCE (Figure 2)

-- PROGRAMMING

A programmer is much more than a technican who encodes a problem solution in machine readable form. A true programmer is deeply involved in formulating the solution to the problem and, in some cases, may even identify and define the problem prior to compounding a solution.

No discussion of programming, therefore, can be limited strictly to an examination of computer coding techniques. The subject must be broadened to include all aspects of the problem and all details pertinent to the problem solution.

-- PERFORMANCE.

We technicians often forget that the criteria for performance measurements are defined by management and may differ greatly between organizations and even between functional areas within an organization. In too many cases we refuse to accept the fact that the only satisfactory solution may require compromise, most often a sacrifice of technical elegance, in order to meet a management objective. A technically advanced solution finished too late may be worthless; one that exceeds the planned cost may be even worse.

Fortunately for us technicians, this discourse will concentrate on technical performance. We will be concerned with traditional indicators, response and throughput, in both batch and transaction processing environments. Many times, one can be gained only at the expense of the other. Luckily, some techniques can improve performance in all instances.

PERFORMANCE CURVE (Figure 3)

A generally acceptable graphic depiction of a machine's performance is a curved line, the "performance curve", showing the gradual performance degradation for the average program as the machine is progressively loaded. In typical transaction processing environments, the curve shows a definite pattern (Curve #1). At first, performance degrades very little as the first few interactive jobs compete with one another. As more jobs are added, each tends to have a

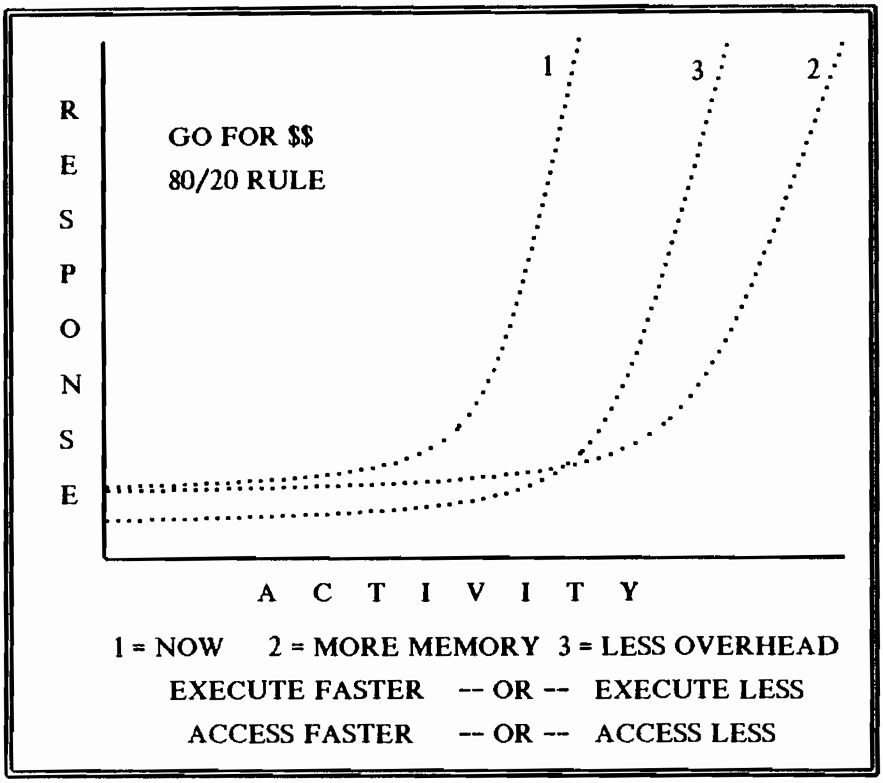


Figure 3

greater negative impact and the curve begins to climb more rapidly. Eventually the "knee in the curve" is reached where each added job causes a disproportionate degradation. At the knee the machine has usually reached the point where the aggregate useful work done by the machine drops for each job introduced.

If the real memory in the machine increases, performance normally improves as shown by Curve #2. On the low load end, the performance improves only slightly if at all since low load performance is not generally memory limited. Mid-range performance improvement is more noticeable and the mid-range itself is extended. The knee still occurs but does not show up until the machine is more heavily loaded. The usual impact of additional memory shows up more in improved throughput rather than in improved individual program response.

Increasing the raw execution power of the machine affects performance differently from increasing its memory. Improvement shows up immediately on the low end of the curve and the knee shows up later. Similar improvement can also be attained by improving mass storage access even without increased raw execution power. Curve #3 shows a typical example of such improvement.

Improved programming exhibits characteristics similar to those of Curve #3. We could expect this since improved programming usually causes less code to be executed or less disc accesses to be made. This, in fact, becomes a basic guideline for improving programming. Most effective performance improving techniques center around reducing executed code and reducing disc access. Since improvements in disc access are more practical to accomplish and will reduce code execution as a byproduct, disc access reduction usually assumes first priority.

TRANSACTION PROCESSING PERFORMANCE (Figure 4)

Batch performance is fairly easily measured. We can easily time how long a job runs, what resources it seems to be absorbing, and how much competing batch jobs inhibit one another. TP performance is much more difficult to quantify.

TP performance is measured by two yardsticks, throughput and response. Throughput is the objective count of the number of transactions that can be processed in any given time period. Response is more difficult to measure because it is a subjective evaluation.

A programmer has more influence on response than on throughput. Additionally, dramatic changes to response may make little change to throughput. This paper will concentrate on response.

TP is too complex to try to examine as a single entity. For simplicity, I am limiting this overview to an evaluation of

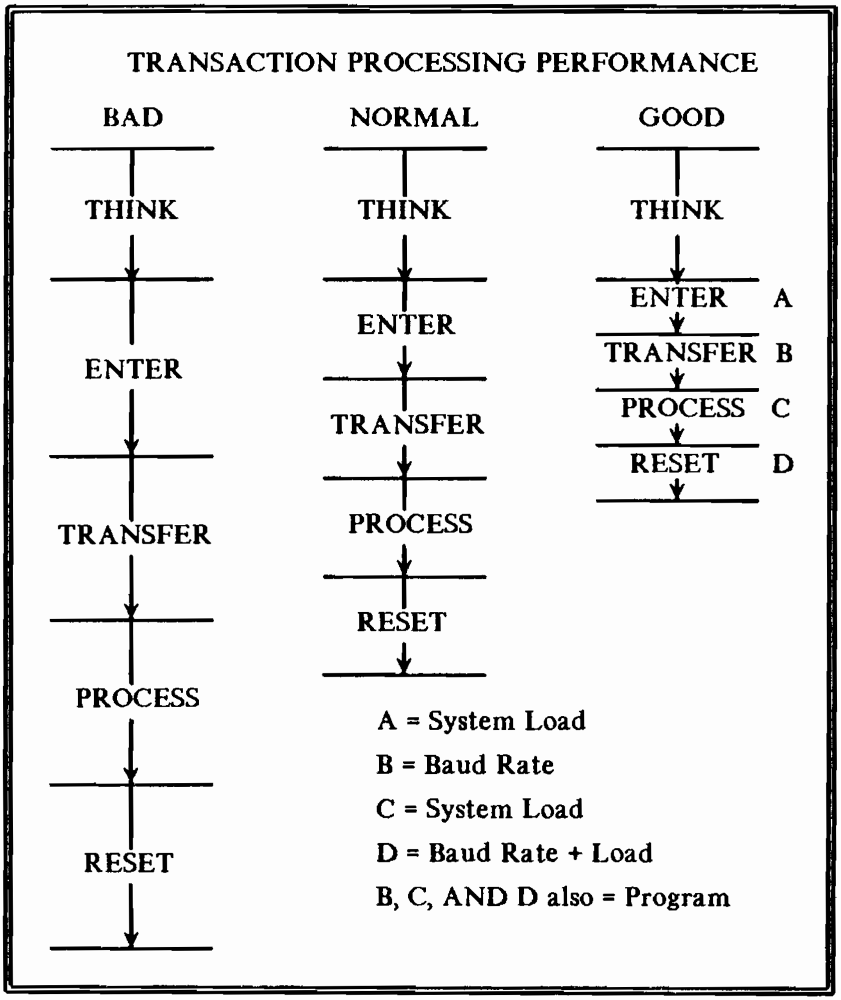

TRANSACTION PROCESSING PERFORMANCE
BAD
NORMAL
GOOD
THINK
THINK
THINK
ENTER
ENTER
ENTER A
TRANSFER B
TRANSFER
PROCESS C
RESET D
TRANSFER
PROCESS
RESET
PROCESS
RESET
A = System Load
B = Baud Rate
C = System Load
D = Baud Rate + Load
B, C, AND D also = Program

Figure 4

the processing of a simple transaction. This can be broken into 5 distinct pieces of time.

1. Think Time.

The time for the user to fill a screen. Think time is normally the longest item in TP and is application dependent. I will arbitrarily disregard think time in this paper.

2. Enter Time.

The time from the hitting of the enter key to the beginning of actual data transfer to the machine. This time is mostly determined by the hardware, the operating system, and the overall machine load. Although a programmer may impact enter time, we will not discuss it.

3. Transfer Time.

The time from the start of data transfer from the screen to its completion. This is mostly a hardware function. In some instances the programmer can change this item. We will address transfer time briefly.

4. Process Time

The time your program spends actually processing the screen input. By definition this is under programmer control. We will concentrate on improving process time.

5. Reset Time

The time required to prepare the terminal for the next user interaction. In pure data entry this time might be trivial. In other applications it might involve screen switches, response transmission or similar lengthy activities. We will examine some cases involving reset time.

Traditional measurements of system response times tend to emphasize the measurement of enter time. For system evaluation this is a valid point of reference. We programmers must concern ourselves with application dependent factors over which we have some control.

Programmers deal most directly with transfer, process, and reset times. These three items are the primary criteria used by users in measuring how responsive a system is.

The user who waits fifteen seconds for the computer to digest a screen of data and prepare the terminal for next input feels that the machine is not responsive. Our job is to make the machine responsive in the eyes of the user.

PERFORMANCE DELIMITERS

DATA

- Stack sizes
- Stack util %
- EDS sizes

CODE

- Segment sizes
- Segmentation
- Seg-seg trans
- Libraries

LANGUAGES

- COBOL
- FORTRAN
- SPL

IMAGE

- Capacities
- MAST vs. DET
- MAN vs. AUTO

V/3000

- Edits
- Enhancements
- Form sizes
- Downloading

SYSTEM TABLES

OTHER FACTORS

Figure 5

PERFORMANCE DELIMETERS (Figure 5)

Almost anything can impact performance. I have consciously oversimplified the situation by dividing the subject into seven groupings. Each of these groupings could be considered most important by any individual. This would be influenced considerably by a person's experience and background. A quick overview of each will set the tone for the presentation of my views on the subject.

GROUPS 1 AND 2 -- DATA AND CODE.

These related items are very important factors influencing performance. They have also received much attention by many people.

This is fortunate in that the programmer has definite guidelines to follow to try to avoid creating totally unacceptable programs. It is unfortunate in that we often assume that the volume of verbiage on a subject indicates its relative impact on our work. I feel this is not always true and that some of us have become entirely too concerned with data and code considerations.

I am almost certain that neither data nor code are the primary culprits when the first questions I am asked about a program are "Do I have my stacks small enough?" or "Have I segmented the program properly?". All too often these are dead giveaways that the programmer has become intimidated by the massive documentation about code and data and has failed to get a good perspective on the whole situation. How, for example, could code and data be causing 10 second response delays in a program that makes 250 data base accesses per response?

I firmly believe that code and data can significantly impact performance. I believe even more firmly that they should be held suspect only after many of the other possible contributors have been reviewed and evaluated.

GROUP 3 -- LANGUAGES.

This is another area where I feel we have all expended too much effort. Except for special cases in which specific capabilities of a language are required, I have yet to find a program which would be meaningfully more efficient in one language than another. The one rule I would accept would be the prohibition of interpretive BASIC in a production environment.

I have seen many instances where the absolute requirement to have a "COBOL shop" has denied a programmer access to efficiencies available in other languages, particularly SPL. I have seen even more cases where the fear of the assumed inefficiencies of COBOL has resulted in FORTRAN or SPL programs which quite often run slower than a COBOL version.

Although this may represent a minority viewpoint, I claim that COBOL will often be the most efficient and the most effective language in the typical commercial application. And even if this may not be true, I further claim that the choice of languages does not have a meaningful impact either way. If you like COBOL, use it. If you hate COBOL, use FORTRAN or perhaps SPL. In any case, use whatever helps you as an individual to get your job done.

GROUP 4 -- IMAGE

By now, everyone should be wondering just what I think are the important performance delimiters. We've finally hit one. IMAGE has a big effect on performance.

The factors sublisted in this group are the ones usually quoted as being critical. These and the use or non-use of sorted chains have been discussed to death. In addition, with the probable exception of the constraints on sorted chains, their importance is generally blown all out of proportion.

We will cover the impact of IMAGE but it will be done from a different vantage point. Our primary concern will be centered around the ways we have structured our data bases and the effects those structures have on performance.

GROUP 5 -- V/3000.

V/3000 is very good but it is not perfect. As with any highly generalized package, V/3000 can probably be "beaten" in any given application by a highly skilled programmer. In some applications, V/3000 may also be a less desirable choice because more performance oriented but less generalized techniques are available. A case in point might be the data capture environment where we would want to consider using the more efficient data capture intrinsics.

Just because V/3000 may not always be the best solution in all instances should not become a rationale for avoiding V/3000 altogether. V/3000 simply has too many capabilities to be ignored. It is also more efficient than most of us probably realize.

V/3000 was introduced with two specific faults which gave some of us some bad memories. Both of these have been corrected. The KSAM oriented forms file structure has been replaced by a vastly superior file access method. The ability to recompile only modified forms has greatly reduced development and maintenance overhead. If we have not reviewed our evaluation of V/3000 since its introduction we may be cheating ourselves.

There are some capabilities still suspect within V/3000. I claim that in most cases, the culprit is the heavy demands we build into our applications rather than the way V/3000 handles those demands. I strongly suspect that most of us would be pleasantly surprised at the performance V/3000 gives compared to that provided by user written code performing the same functions. We would also probably be appalled by the volume of code we would have to write and maintain to replace standard V/3000 capabilities.

There is much potential benefit for us if we closely evaluate our techniques of using V/3000. In many cases we can improve performance by using V/3000 differently. In the average application, however, we would probably do much harm by trying to avoid or replace V/3000.

GROUP 6 -- SYSTEM TABLES.

Let's handle this fast. Look at system tables from an overall system point of view. Forget about them in applications programming.

GROUP 7 -- OTHER FACTORS

When somebody ends a list of items with a group called "other factors", they probably plan to quickly dismiss those same factors as relatively unimportant. In our case, I have deliberately placed them last for emphasis. What many might consider relatively unimportant are the very items experience has taught me to look at more carefully. I think that look will be most revealing.

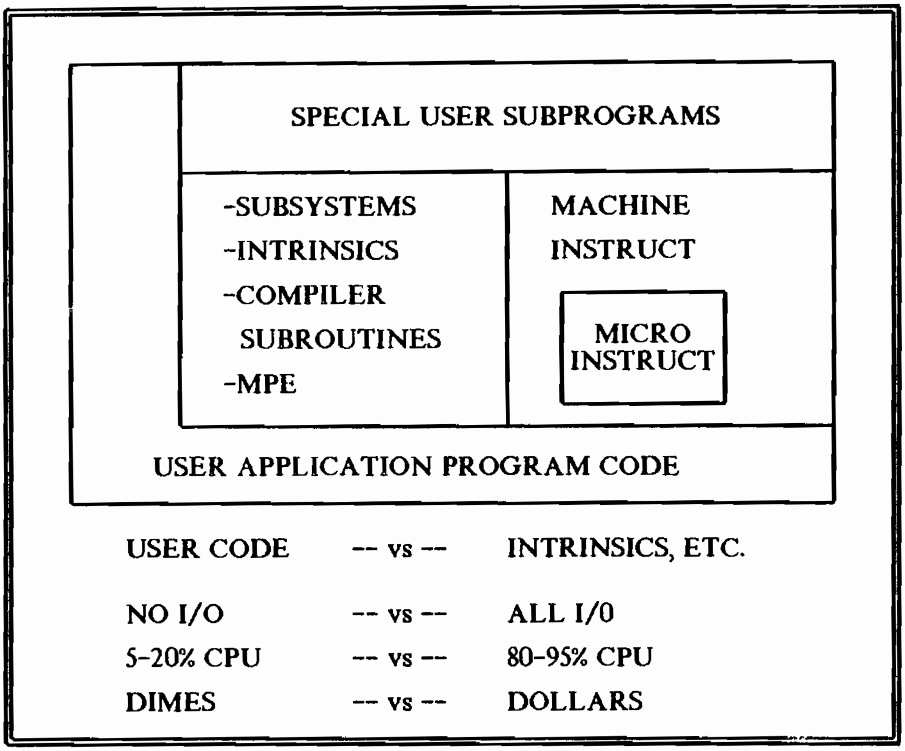

SPECIAL USER SUBPROGRAMS
-SUBSYSTEMS
-INTRINSICS
-COMPILER SUBROUTINES
-MPE
MACHINE INSTRUCT
MICRO INSTRUCT
USER APPLICATION PROGRAM CODE
USER CODE -- vs -- INTRINSICS, ETC.
NO I/O -- vs -- ALL I/0
5-20% CPU -- vs -- 80-95% CPU
DIMES -- vs -- DOLLARS

Figure 6

PROGRAM CHARACTERISTICS (Figure 6)

How we visualize our programs can have a great influence on our attempts to improve their performance. A reasonably accurate picture of a typical commercial application program might surprise some of us. It might also help explain why two programs, one written by a highly technical programmer and the other written by an experienced but relatively non-technical programmer, can have maddeningly (to the technical programmer) similar performance characteristics.

The most important fact to be realized is that the normal application program written on a typical modern commercial computer is a "driver". Whenever we write a program, particularly in a high-level language, we do not generate computer instructions. We are actually writing compiler instructions which will be converted into computer instructions. Our choices of coding techniques can have significant influence on some of the code generated. In general, however, the problem being coded has a far greater effect than our choices of how we code our solution.

A consequence of the "driver" aspect of our programs is that our program code (that is, the portion controlled by our coding techniques) never performs I-O. Except for rare, highly specialized, privileged mode applications, all I-O is performed by MPE Intrinsics. Once we have designed our application characteristics, the I-O required is essentially independent of our programming language or our programming techniques.

Some programmers could be somewhat discouraged at being told their coding techniques have relatively little effect on a program. Others will be relieved on hearing the same message because it allows them to code without fearing that they might mess up a program through "poor" programming. Both of these types of programmers have missed the boat on performance.

Better programmers reprioritize their efforts away from mere technical coding competence and concentrate on design. They realize that the characteristics designed into an application are the major determinants of performance.

Being better programmers, we will concentrate on design in our search for improved performance. Even relatively small changes in design can have more effect than massive changes in pure coding. Wise decisions during application design can have immense impact on eventual performance.

SUBPROGRAMS (Figure 7)

Utilizing subprograms allows us to program for performance. Period. End of sentence.

UTILIZE SUBPROGRAMS !

-- SPEEDS DEVELOPMENT

-- SPEEDS TESTING

-- SPEEDS MAINTENANCE

-- PERMITS EVALUATION
AND
MODIFICATION

Figure 7

A non-trivial program written using subprograms will be faster to develop and test than an equivalent program written as a single unit. This is not prejudiced conjecture by me; it is a widely recognized fact. Since development and test time for a program are included in the broad definition of performance, programs written in subprogram form give improved performance.

Maintenance is also simplified for programs built from subprograms. It is easier to determine where to modify a subprogram than where to modify a unit program. The validity of the change is more easily tested in a subprogram. Subprograms even require less expense to test than do unit programs. Smaller listings, shorter compilations, and more controllable logic give us tremendous return for our investment.

Subprograms would justify themselves solely on the merits claimed up to this point, but we should look deeper. Subprograms can be powerful tools in the attempt to improve application performance.

Subprograms are prime examples of modular program and application architecture. Modularity isolates functions so that modifications affecting those functions can likewise be isolated. Isolation of modifications allows more accurate evaluation of the effects of those modifications. The more accurately we evaluate our modifications, the more effective our modifications can become.

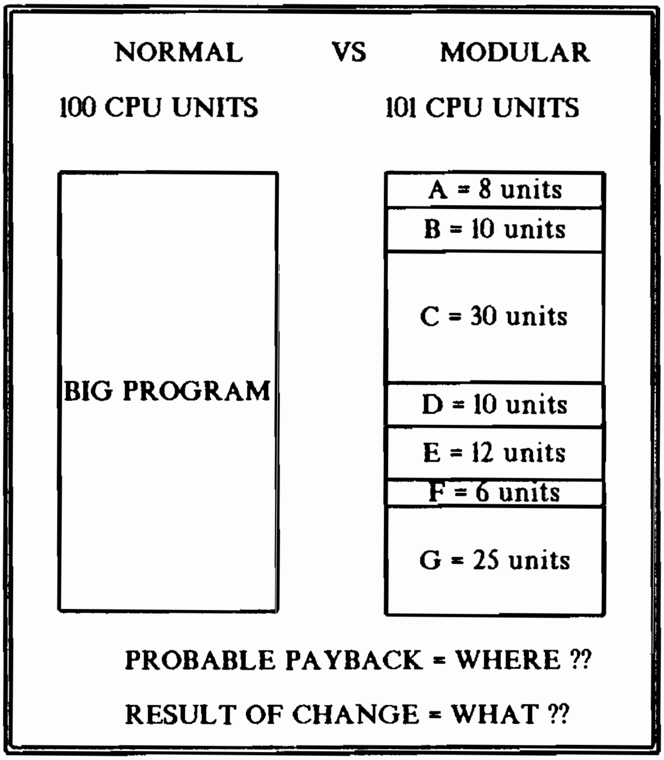

NORMAL
VS
MODULAR
100 CPU UNITS
101 CPU UNITS
BIG PROGRAM
A = 8 units
B = 10 units
C = 30 units
D = 10 units
E = 12 units
F = 6 units
G = 25 units
PROBABLE PAYBACK = WHERE ??
RESULT OF CHANGE = WHAT ??

Figure 8

MORMAL vs MODULAR (Figure 8)

Subprograms do not give us more efficient programs. All factors being equal, a unit program will execute more efficiently than one written in subprogram form. Subprograms, however, give us the means to improve performance and efficiency in ways generally unavailable in unit programs.

Improving performance comes only partly from improving programming. Far more important than how much we improve programming is where we decide to try to improve programming. I would much prefer to reduce a high overhead item than a low one.

Intelligent attempts to improve performance require a disciplined plan of action. That plan must include at least four discreet items:

-- Evaluation of existing performance
-- Identification of candidates for alteration
-- Selection and implementation of changes
-- Evaluation of resultant performance

To get a feel for the potential importance of subprograms for performance improvement, we can review two hypothetical cases. The first involves a unit program; the second, a modular program. In both cases, someone with clout has decided that the performance needs improvement.

CASE I -- THE UNIT PROGRAM.

The programmer follows a rational plan:

-- Measures performance of unmodified program. 100 CPU units are needed for control run.
-- Uses past experience to identify probable bottlenecks
-- Program changes, all quite valid, to improve performance at selected areas
-- Measures performance of modified program. 75 CPU units are needed for test run.

Now we have a few questions to answer:

Q. Was 25 CPU unit improvement good, fair, or poor?
A. Can't tell.

Q. How much of the potential improvement was realized?

A. Can't tell.

Q. Did every change improve performance? A. We don't know. We couldn't make evaluations of each change because the compile cost was too high.

Q. What do we try next? A. Whatever the boss says.

This would have been so easy had the original changes taken us from 100 CPU units to 20. That type of improvement gets praised, not questioned. But who said everything was easy?

CASE II -- THE MODULAR PROGRAM.

This programmer also follows a rational plan:

-- Measures performance of unmodified program. 101 CPU units are needed for control run. Usage per module ranges from 6 to 30 units.
-- Decides that most probably payback is in modules C and G with 30 and 25 unit loads, respectively
-- Uses past experience to decide that module G is most likely candidate.
-- Program changes to module G
-- Measures performance of modified program. 86 CPU units are needed for test run. Module G has gone from 25 units to 10 units.

This programmer also must answer some questions:

Q. Was a 15 CPU unit improvement good, fair, or poor?
A. Quite good. The portion changed showed a 60% (from 25 units to 10) reduction.

Q. How much of the potential improvement was realized?
A. For module G, probably most of it. But we only changed 25% of the program. 75% still merits evaluation.

Q. Did every change improve performance? A. Looks probable.

Q. What do we try next? A. Module C is a likely candidate because it absorbs 30 units during execution. That's more than a third of the remaining overhead!

We may require a number of iterations before we reach the point of diminished returns. At least we have a better way to tell where we have probably reached it. In addition, we have begun to build a body of experience to help us optimize our next program more easily.

SECTION 2 BIG ITEM CHECKLISTS

Talking about techniques to use in isolating performance problem areas is valuable. Knowing what to do next is equally important. This paper will attempt to identify some of the most frequent "next steps".

Sometimes, of course, the "next step" becomes the only step. This occurs most often when we are called upon to help optimize a unit program whose performance is suspect. We can't waste time wishing the program could be more easily analyzed. After all, if it were easily analyzed, we wouldn't have been called in. So we take it as we find it.

Every experienced performance consultant has a mental checklist of potential problem areas. This checklist includes specific techniques found helpful in the past and the expected benefits for each.

Each checklist is different. The differences depend upon the consultants background, track record, and personal biases. A specialist experienced in commercial applications has a different checklist from a specialist who has worked with technical applications. Similarily, checklists based on batch applications will differ from those written for on-line systems.

My background is in commercial applications, both batch and on-line. I would like to share part of my checklist with you. The sequence is for convenience and continuity; it has no priority implications. For each major item, I will organize its analysis this way:

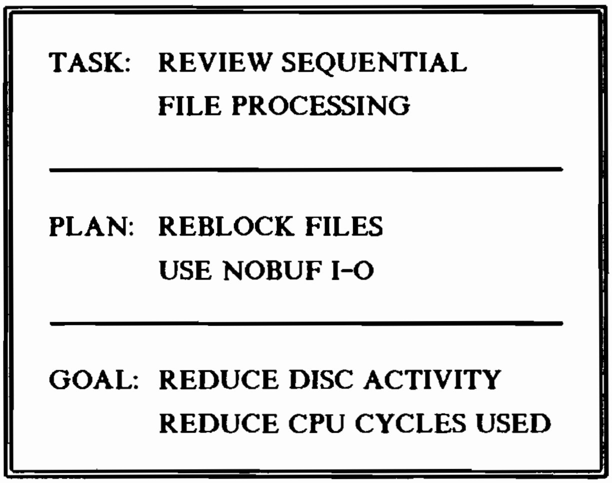
-- A header visual showing:

  - TASK: The description of the checklist item.

  - PLAN: The proposed corrective action most likely to succeed.

  - GOAL: The expected benefits.

- -- One or more subordinate visuals showing.
  - Why the item might be degrading performance.
  - Detail examination of the proposed action.
  - Why the proposed action should improve performance.



TASK: REVIEW SEQUENTIAL
FILE PROCESSING

PLAN: REBLOCK FILES
USE NOBUF I-O

GOAL: REDUCE DISC ACTIVITY
REDUCE CPU CYCLES USED

Figure 9

SEQUENTIAL FILE PROCESSING (Figure 9)

Almost every shop has batch programs that process large sequential files. Quite often they are programs originally written for Brand-X and converted to run on the 3000. They frequently run much slower than we think they should.

We keep reminding ourselves that most commercial applications are I-O bound, not CPU intensive. We begin wondering about the power of the 3000 when our "I-O Bound" programs run at close to 100% CPU utilization. It's time we found out why this happens.

TASK: Check out sequential files looking for high record volumes, low blocking factors, and default file access.

PLAN: Increase blocking factors and replace default file access with nobuf I-O.

GOAL: Reduce disc activity (blocking factors). Reduce CPU load (nobuf I-O).

I-O -- BUFFERED vs. NOBUF (Figure 10)

Those of us who learned programming on a Brand-X machine, know how sequential files are processed. A pair of buffers are set up in the program and the physical I-O system tries to keep them full. The logical I-O system provides records to us by indexing through the buffers as I-O is requested.

This indexing through internal buffers is extremely efficient. Machines using this approach to sequential processing usually handle batch processing better than on-line processing.

Every operating system on every computer is optimized for a particular environment. This includes both the internal architecture of the I-O system and the choice of standard defaults for its user interface. HP emphasizes on-line processing and has designed its I-O system accordingly. Batch processing is performed well but suffers somewhat to benefit on-line work.

On-line processing emphasizes random retrieval. Random retrieval implies retrieval of a record from a reasonably

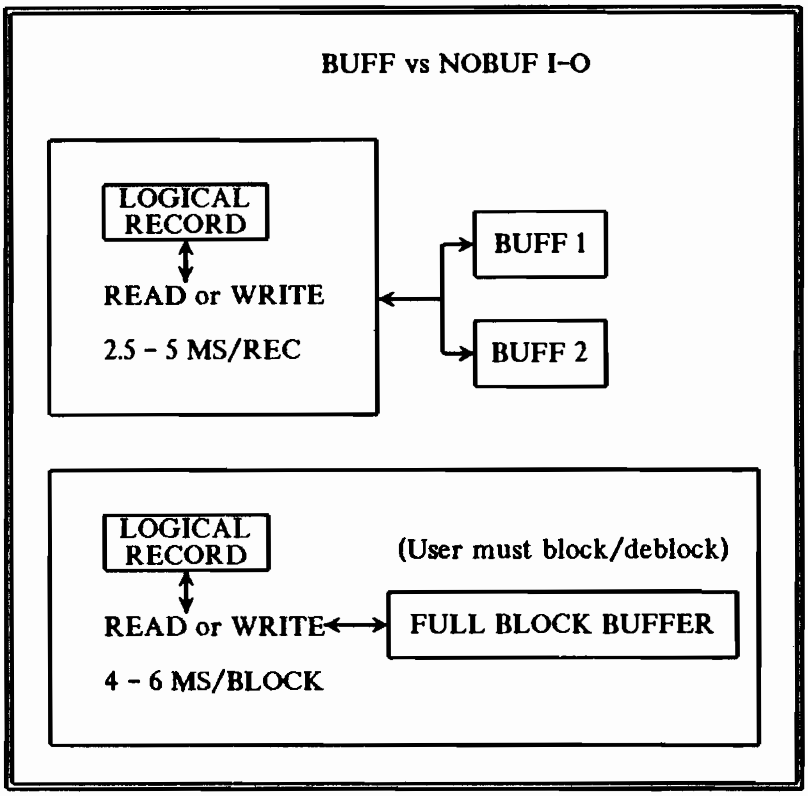

BUFF vs NOBUF I-O
LOGICAL RECORD
READ or WRITE
2.5 - 5 MS/REC
BUFF 1
BUFF 2
LOGICAL RECORD
(User must block/deblock)
READ or WRITE
FULL BLOCK BUFFER
4 - 6 MS/BLOCK

Figure 10

well-defined location on disc. Knowing a records location eliminates most of the benefits of large blocking factors. Therefore, HP has set up relatively small default blocking factors. These utilize disc space well but are usually comparatively small for batch processing.

On-line processing also requires effective file sharing capabilities. Files are quite difficult to share when the I-O system puts the buffer inside programs. HP simplifies file sharing by isolating the buffer from the program. This is excellent for on-line activities but increases overhead when doing batch.

File buffers reside in extra data segments under MPE. Logical I-O requires the file system to expend considerable effort to transfer records back and forth between the user stack and these extra data segments. This explains the high CPU load during sequential file access.

Once we know how the defaults in MPE increase overhead in batch applications we can make intelligent adjustments. The rewards are well worth the effort. In typical cases, we can reduce overhead by 70 to 90 percent.

Blocking factors are easily changed by coding the "REC" parameter in the "BUILD" command. I can't tell anyone what factors to use but I would probably choose between 7 and 30 depending on record size.

Conversion to nobuf I-O is not so simple but contributes most to CPU load reductions. There are two basic ways to do this:

-- Code your routine directly into your program. Although this is how I coded my sample program, I prefer the second technique.

-- Code your routine in a subprogram. I prefer this technique. It suits my mode of operation.

RECORD SORTING (Figure 11)

Every shop needs sort capabilities. Batch applications are particularly heavy users of sorts because they are inherently sequence dependent.

Sorting places heavy demands on the machine. Although we cannot reasonably eliminate sorting from our programs, we have techniques to reduce their overhead.

TASK: Review our use of the sort capability in our environment.

PLAN: Avoid file-to-file sorts and the use of the stand-alone sort subsystem.

TASK: REVIEW USE OF
SORT SUBSYSTEM

PLAN: AVOID SORT SUBSYSTEM
AVOID FILE-FILE SORTS

GOAL: REDUCE DISC ACTIVITY
REDUCE CPU CYCLES USED

Figure 11

GOAL: Reduce I-O activity in sort functions. As a secondary benefit, reduce CPU load on the system.

BRUTE FORCE (Figure 12)

When you had a small machine and tape was your primary storage media you learned how to drag data through programs using brute force. The standard mode for sorting was to read a tape into the sort and write sorted records back to tape. Sometimes this had to be done in multiple passes with multiple tapes. It wasn't much but it certainly beat loading and unloading card hoppers.

Modern computer systems support and use tape but they rely more often on disc for primary storage. Modern programmers have also begun to rely on disc. They have finally gotten rid of their little drawers full of punched cards. Why, then, do they still do their work, particularly their sorts, by brute force.

The stand-alone sort is useful and necessary. It is also a resource hog. The I-O required for a sort is extensive, especially when added to the I-O to read and write output

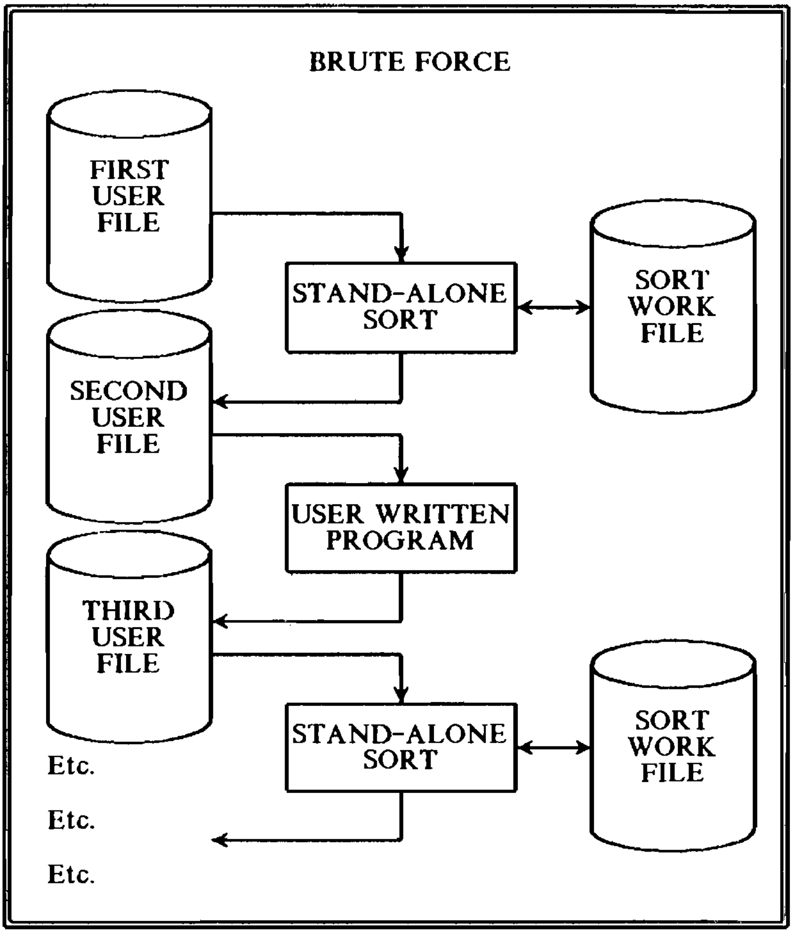

BRUTE FORCE
FIRST USER FILE
STAND-ALONE SORT
SORT WORK FILE
SECOND USER FILE
USER WRITTEN PROGRAM
THIRD USER FILE
STAND-ALONE SORT
SORT WORK FILE
Etc.
Etc.
Etc.

Figure 12

files.

Disc availability is a prime constraint on performance. Sorts, particularly the stand-alone file-to-file sort, eat deeply into this availability. In the interest of performance we should try to reduce these activities whenever practical.

FANCY BRUTE FORCE (Figure 13)

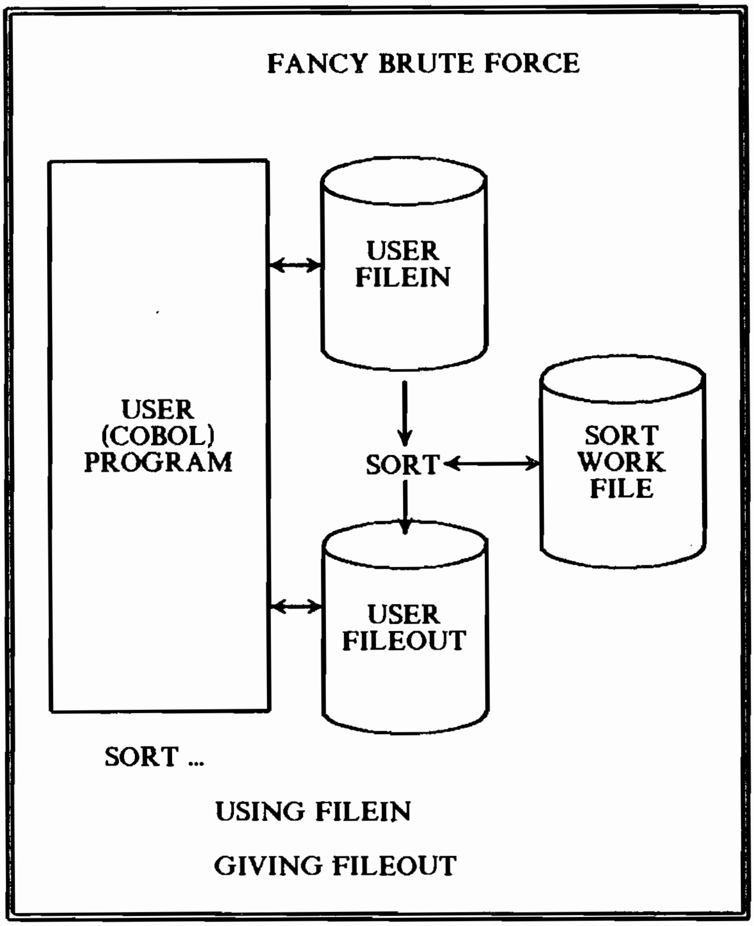

FANCY BRUTE FORCE
USER (COBOL) PROGRAM
USER FILEIN
SORT
SORT WORK FILE
USER FILEOUT
SORT ...
USING FILEIN
GIVING FILEOUT

Figure 13

Programmers have learned how to invoke the sort programmatically. They have effectively used this to replace two or more programs and one or more sorts with single programs. Then they have held back from using the true capabilities of programmatic sort access.

Programmatic file-to-file sorts are not necessarily bad but they usually have a negative effect on performance. They use up disc resources at at least the same rate as stand-alone sorts. Except for very small data volumes, programs written with file-to-file internal sorts usually run slower than the individual programs and sorts they replaced.

Some shops will not allow programmers to use sort capabilities programmatically. I think they are missing a good opportunity by this blanket condemnation. On the other hand, if they are only avoiding "fancy brute force", they can be partially excused.

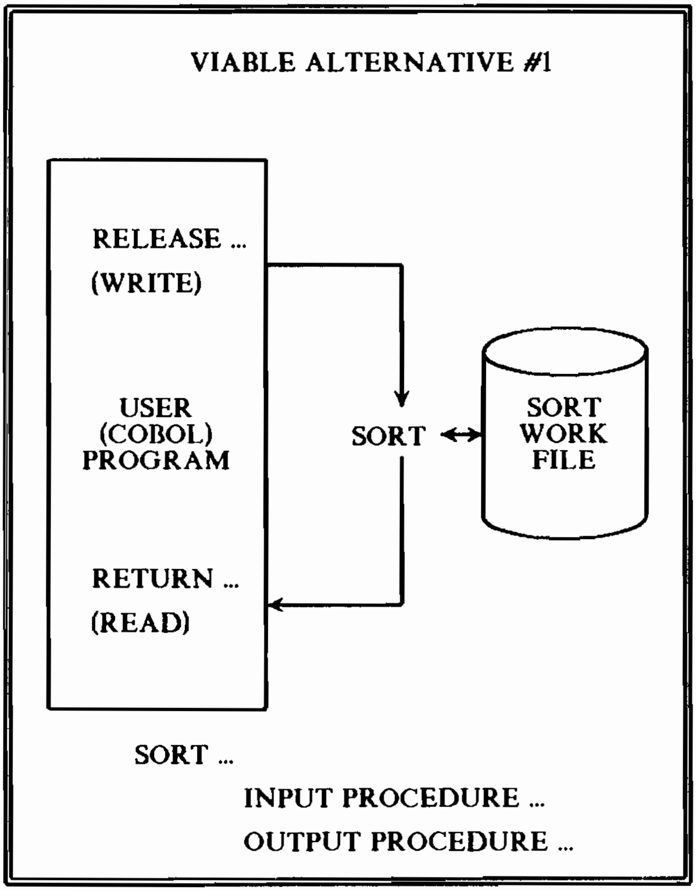

VIABLE ALTERNATIVE #1
RELEASE ...
(WRITE)
USER (COBOL) PROGRAM
RETURN ...
(READ)
SORT
SORT WORK FILE
SORT ...
INPUT PROCEDURE ...
OUTPUT PROCEDURE ...

Figure 14

VIABLE ALTERNATIVE #1, DIRECT SORT INTERFACE (Figure 14)

Programmatic access to the sort subsystem gives the programmer many attractive options. Hooks built into the sort allow programmers to pass records directly to the sort and receive sorted records directly from the sort. This capability can be used effectively to improve program performance.

Direct interaction with the sort allows us to avoid disc activity. Every time we interact directly with sort we avoid two potential disc accesses; we have eliminated a read access and a write access. This can greatly reduce disc activity.

We also reduce CPU overhead by talking directly to the sort. We get rid of the CPU overhead for the file system to process our logical I-O. Unfortunately, there is a price to be paid for this ability.

Interaction with the sort requires direct resources other than disc I-O and CPU cycles. The sort needs memory to be efficient and that memory comes from the user stack. If our stack is too small or the program data is large the internal sort may lose its value. It may then become a burden.

Another limitation of the internal sort is the inability to have multiple sorts executing simultaneously. In programs with multiple internal sorts we may have to allow some file-to-file sorts or the logical equivalent.

VIABLE ALTERNATIVE # 2, PROCESS HANDLING (Figure 15)

MPE offers us another means to reduce sort overhead. We can use process handling as a means to bypass I-O in sorting applications.

Some shops fear process handling. I wish more of them could begin using it to advantage. Perhaps there is too much emphasis on the "special" in "special capabilities". Whatever the reason, many of us look upon process handling as a tool only for exceptional cases. We should view it as an exceptional tool useful in many applications.

There is no justification for reserving process handling

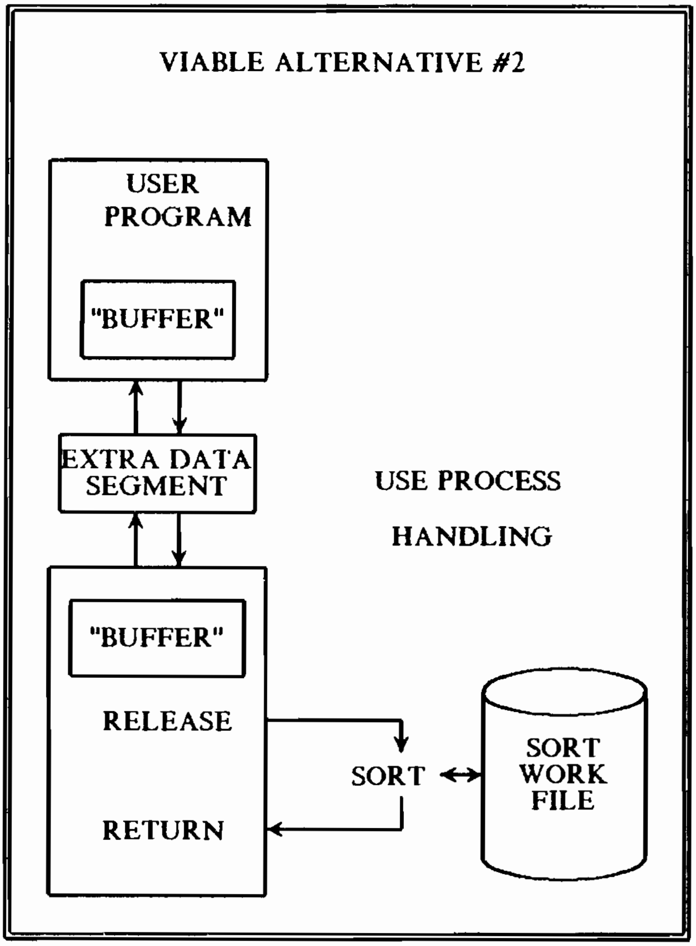

VIABLE ALTERNATIVE #2
USER PROGRAM
"BUFFER"
EXTRA DATA SEGMENT
USE PROCESS HANDLING
"BUFFER"
RELEASE
RETURN
SORT
SORT WORK FILE

Figure 15

solely for esoteric or multi-threading environments. Process handling is perfectly suitable for use in relatively simple, single-threaded applications. An obvious use would be in a program involving sorts where we can use multiple processes to advantage.

We should examine how sort functions can be accomplished using process handling. The concept is similar to the normal programmatic sort interface especially if we isolate all calls to the process handling intrinsics in a subprogram. The rest is simplicity itself.

1. The master program initiates sorting by a call to the subprogram. The subprogram creates and activates a slave program whose job is to sort records. The subprogram also creates an extra data segment to use in passing blocks of records to the slave. It waits until the slave program is ready for work.

2. The slave begins life by starting up its own internal sort. Its subprogram then wakens the master, in effect saying "OK, I'm ready". It waits for instructions.

3. The master sends raw records to the slave for sorting. Every call to the master subprogram represents logical passage of a record to the slave. The subprogram fills an internal buffer with records.

4. When the subprogram has a full buffer it loads it into the extra data segment and wakes the slave, effectively saying "OK, give these to sort". It waits until the slave has done its work.

5. The slave retrieves the extra data segment, unloads the logical records, and passes them to the sort. When finished, it wakes the master, saying "OK, I'm ready for more". It waits for more.

6. Steps 3, 4 and 5 are looped through until all records have been sent to the slave. The master then sends an "end of data" message to the slave after preparing to receive sorted records. The master waits now.

7. The slave receives the "end of data" message and lets actual sorting begin. When sorted records are available, the slave is ready to awaken its master.

8. The return of sorted records from master to slave is a reversal of the process described for sending unsorted records from master to slave.

A sample program is included with this paper to show the technique. It does not use subprogram interfaces because I wanted to isolate the example in one source file. In real life, I would recommend a subprogram.

This example obviously reduces disc activity. It offers other benefits as well.

1. The slave has its own stack which is not loaded with application data. The sort can be given plenty of room to breathe. The slave can perform actual sorting more efficiently than could its master.

2. The program is no longer limited to having only one sort active at any time. Except for the normal constraints of MPE, any number of sort slaves may be active at any time.

As with other alternative techniques, process handling extracts a cost.

1. Process handling absorbs CPU overhead. If records are not passed back and forth in blocks, the overhead may be relatively high.

2. Multiple processes and their stacks need memory. This may cause problems in some cases. Your machine size and workload profile are critical decision criteria.

3. Process handling is not difficult but is more complex than the use of standard compiler features. Subprograms eliminate this problem once you have the subprograms written and tested.

DATA VALIDATION AND CONVERSION (Figure 16)

Both batch and on-line programs perform extensive data validations and conversions. Quite often these account for a high portion of the overhead within a program.

Validations and conversions come in two basic flavors-algorithmic and tabulated.

-- Algorithmic validations check validity according to a processing rule. Check-digit calculations and pattern

TASK: REVIEW VALIDATIONS
VIA FILE ACCESS

PLAN: REPLACE FILES WITH
PB-RELATIVE CODE

GOAL: ELIMINATE DISC ACTIVITY
REDUCE CPU CYCLES USED

Figure 16

matches are examples.

-- Algorithmic conversions convert data based upon a conversion algorithm. Julian to Gregorian data conversion is an example.

-- Tabulated validations check validity by searching a table or file for a "hit". Customer validation through attempted retrieval against a data base is an example.

-- Tabulated conversions convert data from argument to result by searching a table or a file. Converting numeric error codes to meaningful error messages is an example.

Algorithmic techniques and table oriented searches are normally efficient but may be difficult to modify and maintain. File oriented techniques are easily modified but absorb considerable overhead. Performance considerations may make file oriented techniques too expensive.

TASK: Review file oriented validations and conversions.

PLAN: Replace files with PB-relative code. This normally takes the form of a binary search procedure.

GOAL: Eliminate disc I-O completely when possible. Significantly reduce CPU overhead.

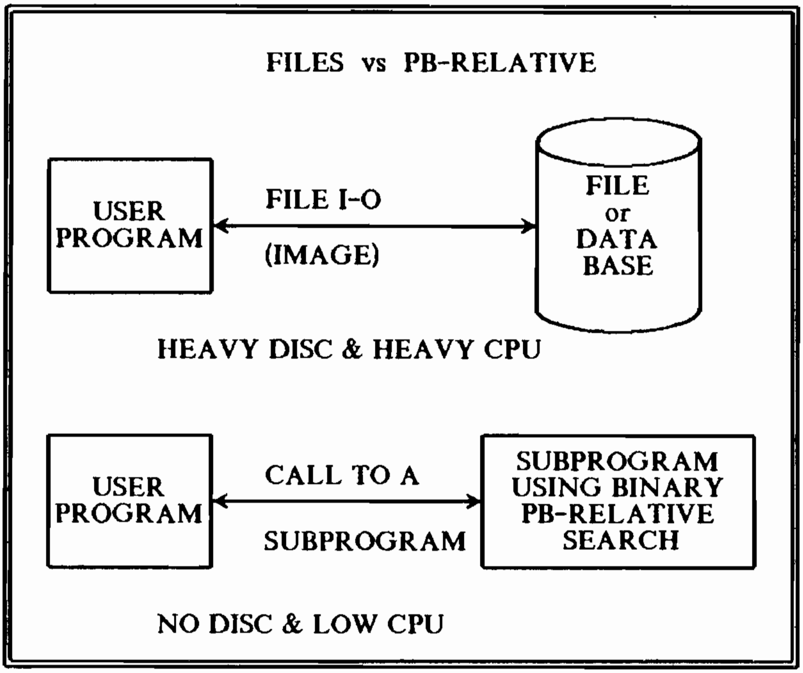

FILES vs PB-RELATIVE
USER PROGRAM
FILE I-O
(IMAGE)
FILE or DATA BASE
HEAVY DISC & HEAVY CPU
USER PROGRAM
CALL TO A
SUBPROGRAM
SUBPROGRAM USING BINARY PB-RELATIVE SEARCH
NO DISC & LOW CPU

Figure 17

FILES vs PB-RELATIVE (Figure 17)

Files are frequently used for data validation or code expansion in all data processing. We usually use KSAM or IMAGE. In either case the disc I-0 is extensive and involves considerable overhead.

In many cases there is no viable alternative to file access. Volatile information is ideally suited to randomly accessed and maintained file structures. Massive quantities of data cannot be economically maintained except on disc media.

In other cases we may be able to use techniques that require little or no disc access. The following are the normal techniques used.

1. Tables may be hard-coded into storage if the information is not too massive and is not volatile. Massive data volumes are prohibitive and volative data values create maintenance hang-ups.

Computer Museum

2. Volatile tables may be filled at execution time via file access. This is impractical for large files and may cause excessive overhead in short duration programs.

3. Values may be hard-coded into program code as literals. This saves stack but creates maintenance problems for volatile values.

4. Data values may be loaded into extra data segments. This is especially useful for highly volatile data values in transaction processing applications utilizing multi-threaded process handling. It is a relatively complex approach but can be of great use.

Another available but seldom used technique is to place data values into PB-relative code. The code is SPL and the internal retrieval is via a binary search. This technique has tremendous potential for improving performance.

1. Disc access is eliminated.

2. CPU overhead for retrieval is extremely low.

3. Storage demands are relatively low, particularly if the code resides in readily shareable SL segments.

4. Access is simple since a single call statement is sufficient.

Nothing is free. For all its benefits, PB-relative code has many drawbacks.

1. Dynamic changes cannot be made to PB-relative code.

2. SPL is a requirement for PB-relative binary access.

3. CST table limitations may restrict use of easily shareable SL files.

On balance, I strongly recommend PB-relative techniques in cases where performance potential is needed. I also recommend that they not be applied just because they are available. Like many other techniques, its use must be based upon its relative value within the application.

Two example subprograms are included in this paper. One demonstrates retrieval of fixed length values; the other, variable length. Obviously, either will validate data arguments.

PB-RELATIVE MAINTENANCE (Figure 18)

A major reason to avoid using PB-relative code for validation and conversion purposes is the difficulty it presents to maintenance. The potential benefits cannot, however, be ignored.

Maintenance of tabular data in PB-relative code is not trivial but it need not be excessively difficult. The first step requires that records be prepared for batch input to a maintenance program. This is a common function in all batch maintenance applications and should be no problem.

The actual maintenance run is different depending on how we wish to apply the maintenance. There are three primary approaches to applying maintenance.

1. Maintenance data can be converted to CON (constant) constructs in SPL. These can be inserted into a skeleton program which will be compiled into a USL. This is a straight-forward approach but requires multiple steps.

2. Maintenance data can be applied directly to a generalized USL skeleton. This is quite efficient but requires considerable knowledge of USL structure.

3. Maintenance data can be applied directly to either an RL

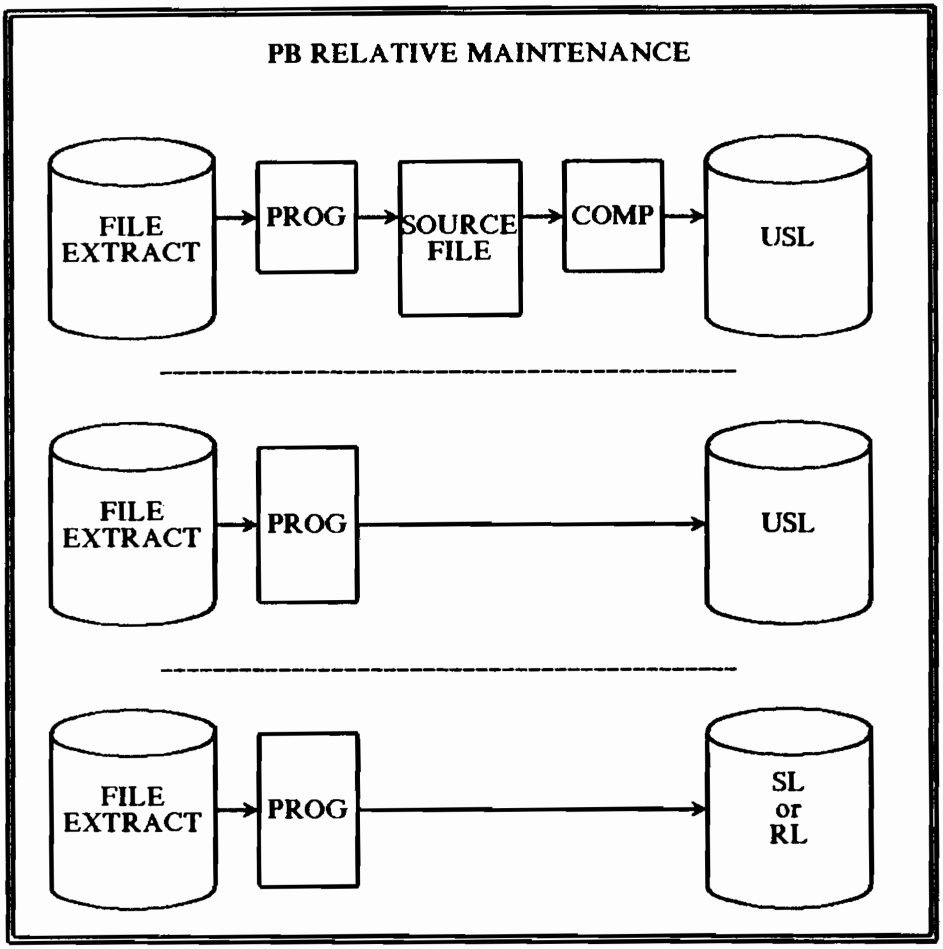

PB RELATIVE MAINTENANCE
FILE EXTRACT
PROG
SOURCE FILE
COMP
USL
FILE EXTRACT
PROG
USL
FILE EXTRACT
PROG
SL or RL

Figure 18

or an SL. This is even more direct than modifying a USL but is probably more difficult.

We have a fourth technique that I hesitated to put on the diagram. We can change the program directly. I left it off the diagram because it destroys my credibility with the fragile types.

Program code is not user-alterable during execution. That does not keep us from altering files just because they happen to represent programs. There are many things you can do with program files if the need arises. Keep your eyes open for opportunities. It's fun.

TASK: REVIEW IMAGE
DATA BASES

PLAN: SIMPLIFY
STRUCTURE

GOAL: REDUCE DISC ACTIVITY
REDUCE CPU CYCLES USED

Figure 19

IMAGE DATA BASES (SIMPLIFICATION) (Figure 19)

Commercial applications contain requirements that often lead to complex data structures. IMAGE gives us the means to control these data structures with reasonable ease. Sometimes we forget that what is reasonable for us from a design and access point of view may be totally unreasonable at the machine performance level.

Designing paths to detail data set records is a simple task, particularly if the path is from an automatic master. The desire for rapid random retrieval often justifies this structure.

Using a path from a detail data set to a manual master can provide non-programmatic data validation as well as potentially rapid retrieval. This capability justifies many of our decisions during application and data base design.

When performance becomes unsatisfactory we have to revalue our decisions. With experience, most of us become more selective with the facilities we include in our designs. We have learned that complex, aesthetically pleasing structures may become our white elephants of performance.

TASK: Review IMAGE data base structures.

PLAN: Revalue our design with hopes of simplifying the structure without undue impact on functional performance.

GOAL: Reduce disc activity and CPU load.

REDUCE LINKAGE PATHS (Figure 20)

IMAGE gives excellent performance in random record retrieval. This is one of the major reasons why IMAGE has become so widely accepted. We appreciate the capability to read detail records by multiple key items. Sometimes we get carried away in our appreciation and go too far.

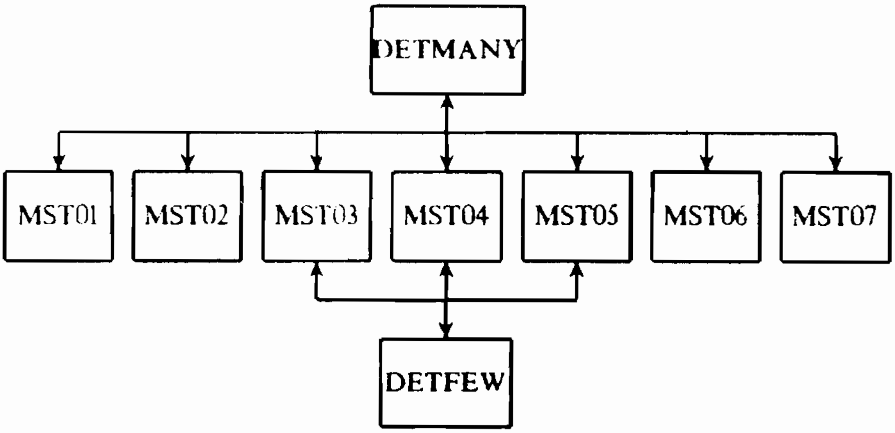
Rapid detail access is achieved using pointers which allow precise record location. These pointers have to be created before the record is accessible. The price of rapid retrieval is paid by the machine when it sets these pointers.

The majority of the overhead for adding a detail record linked to multiple masters comes from establishing the linkages to those masters. On a dedicated Series III you can closely predict that about 7 or 8 linkages can be created or deleted per wall second. This has tremendous implications for performance.

Adding or deleting a detail linked to 7 masters will take about 1 wall second. This is the main reason why IMAGE reloads and batch record maintenance programs run relatively slowly. With 7 linkage paths per detail you can only expect about 4000 adds and deletes per hour.

Transaction processing is also severely impacted by the

REDUCE LINKAGE PATHS



ON DEDICATED MACHINE (III OR 44)

-- DETMANY: Add/Delete will take about 1.0 sec wall time

-- DETFEW: Add/Delete will take about 0.5 sec wall time

Figure 20

maintenance of linkage paths. If a transaction causes additions of 3 records and each has 7 linkage paths you have every reason to expect at least a 3 second delay during those adds Under normal circumstances when others are sharing the machine you should not be surprised with much longer delays. This helps explains the following common situation. You can supply the interpretation.

- Design goals call for 5 second maximum processing time delay

- Testing shows excellent average delays of 3.5 seconds

- Initial production delays average a satisfactory 4.5 seconds

- 3 months later the delays are averaging 10 seconds

- You have a problem. The test cases did not predict the realistic performance.

Worst case performance comes during record modification if a search item needs to be changed. This requires a physical delete and add. The time for this change is the sum of the delete and add times. This type of processing can break the back of an application.

All too often the ability to define multiple paths into details seduces us into defining too many paths. Unless the path is required or gives a high priority extra capability you should think seriously before creating it.

1. Every path defined has essentially the same cost.

2. Every path probably does not give the same payback.

3. Is the payback worth the cost?

4. Will the extra path create the monster called "change = delete + add"?

Limiting the number of linkage will improve performance. Consider the case where we go from 7 to 3 linkage paths.

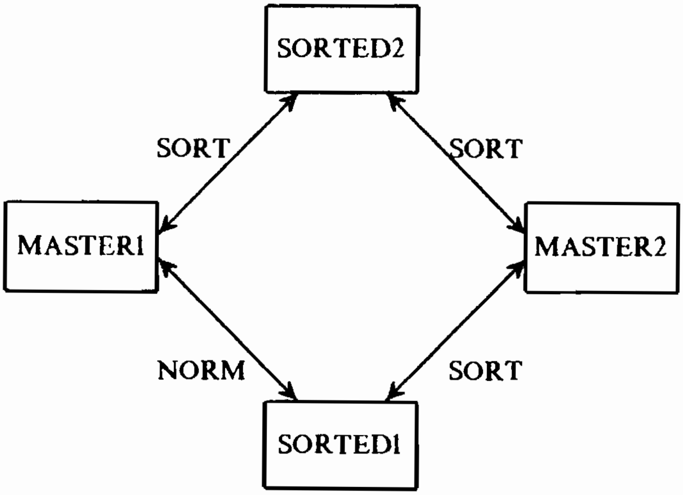
1. Processing delays will drop from 1 to about .5 seconds.

2. Delete and add for change will occur less often and will take only half as long.

3. Delays on a loaded system will probably grow much less and will be less dramatic.

4. Reloads and batch maintenance will be significantly faster.

LIMIT SORTED CHAINS



ASSUMING NON-TRIVIAL CHAIN LENGTHS

-- Add/Delete & access of SORTED2 MUST BE relatively slow

-- Add/Delete & access of SORTED1 MIGHT NOT BE slow

Figure 21

LIMIT SORTED CHAINS (Figure 21)

IMAGE allows details within a logical path to be sorted by some other item value. This is a powerful capability which may be of use to us. It is also an expensive capability.

Addition of a detail with a sorted path causes more overhead because sort sequence must be maintained. IMAGE starts at the logical high end of the chain and follows it backwards until it finds the correct logical home for the detail. It then links the detail into the logical path and goes about its business for the remaining linkage paths.

If the detail fits at the logical end of the chain, the added overhead is trivial. If sort-item values are random, IMAGE must read half through the chain on average to find a logical home. This can be quite expensive.

Note: Sorted paths can give you unpleasant surprises.

1. The value sorted extends from the sort item to the end of the record. This is the "implied sort".

2. Additions will use the entire implied sort value.

3. Any sort before addition should include the full implied sort item. Hideous performance can result if this is not done. This explains why sort items should usually be at the end of a record.

4. Implied sort sequences can be useful. They are also functionally dangerous. IMAGE allows update in place on non-sort items. The implied sequence disappears once updates are done to fields within the implied sort.

Defining multiple sorted paths within a detail guarantees performance degradation. There is no way to avoid extra overhead with multiple sorted paths.

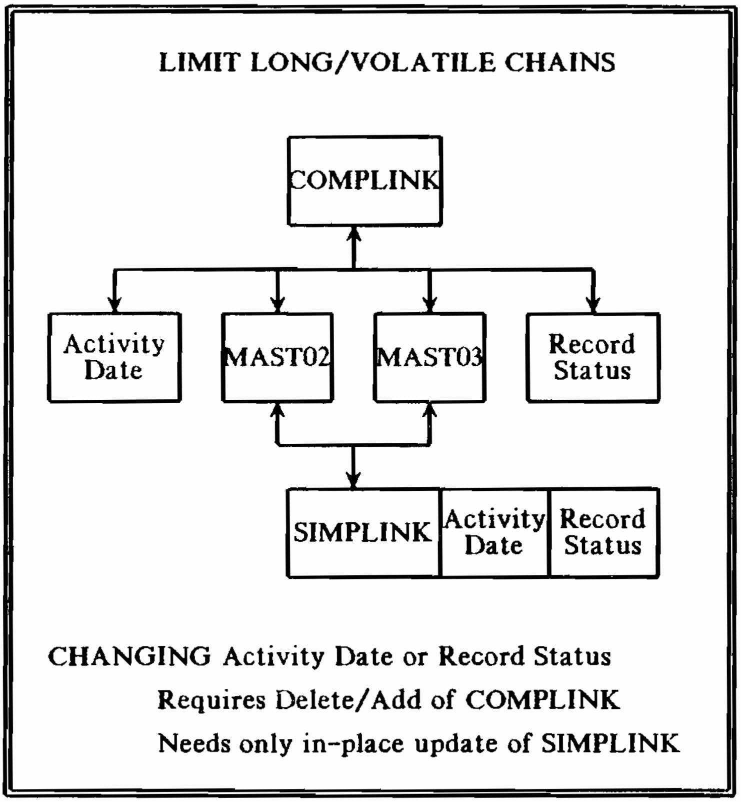
LIMIT LONG OR VOLATILE CHAINS (Figure 22)

The ability to chain logically related records together is a necessary function in any Data Base Management System. This function is designed with relatively short logical chains in mind. It is not meant to be abused.

A good logical chain usually maintains relationships based upon important data items. They are designed to preserve solid logical relationships among records.

1. Customer identities linked to their orders.

2. Line items within an order linked to an order header.



LIMIT LONG/VOLATILE CHAINS

COMPLINK
Activity Date
MAST02
MAST03
Record Status
SIMPLINK
Activity Date
Record Status

CHANGING Activity Date or Record Status
Requires Delete/Add of COMPLINK
Needs only in-place update of SIMPLINK

Figure 22

3. Order line items linked to inventory status records.

4. Inventory status records linked to open purchase orders.

5. Purchase orders linked to responsible vendors.

A poor logical chain usually attempts to maintain relationships which are relatively less important. They are often designed to try to create artificial order out of inherent chaos. Even worse, they may exist simply because the capability to create them is provided.

1. Personnel records linked to employee sex.

2. Invoice records linked to invoice status.

3. Student records linked to grade values.

4. Inventory records linked to last activity dates.

Another form of poor chain would be the case where the search value is volatile. Every time the value changes, a complete delete/add must be done. This can be very expensive.

My generalized definition of a poor logical chain is based upon present technology. A poor chain is one whose cost exceeds its value. When technology reduces cost sufficiently, I will change my definition.

The definition of a poor chain is also not absolute. If your application benefits more from a chain than it spends to have the chain, the chain is good. Good and bad are merely a comparison of relative gain to relative cost.

TASK: REVIEW IMAGE
DATA BASES

PLAN: COMPLICATE
STRUCTURE

GOAL: REDUCE DISC ACTIVITY !!
REDUCE CPU CYCLES USED
INCREASE CAPABILITIES

Figure 23

IMAGE DATA BASES (COMPLICATION) (Figure 23)

Simple IMAGE data base structures generally absorb less overhead per function than do complex structures. This leads us to avoid complex structures to maintain good performance. Sometimes, however, the simple structure becomes a burden and causes unacceptable performance for some required application functions.

Overall performance within an application is a complex entity. Each function performed carries an inherent overhead based upon the function, how often it is performed, and the structure of the data base it accesses. Simple structures can degrade performance if they fail to permit efficient processing of frequently required functions.

Relatively complex structures may benefit overall performance if the structure matches the intended use. Simple structures, while inherently more efficient, may degrade performance if they do not satisfy the application. We cannot judge nor can we design a data base without extensive knowledge of the application.

We may find that our application cannot be serviced satisfactorily without complicating our data base. With proper planning we may be able to use complexity to our advantage to improve both functional and overall performance.

TASK: Review IMAGE data base structures.

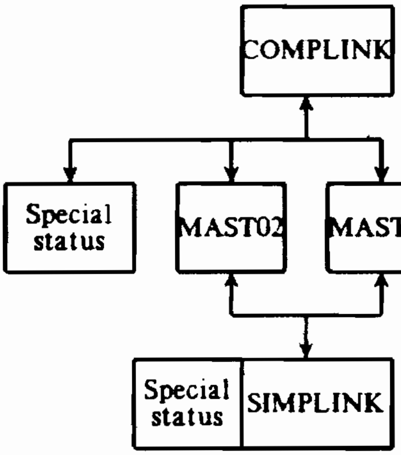
PLAN: Revalue our design based upon our knowledge of the application and try to find places where a more complex structure can improve performance.

GOAL: Selectively increase complexity to make our application overhead go down when performing required functions.

SELECTIVE CONSTRUCTIVE ABUSE (Figure 24)

Simple data structures are normally preferable to complex structures. Chains should usually be avoided where data volatility is a problem. But you can throw out any generalized rule if it serves you poorly.

SELECTIVE CONSTRUCTIVE ABUSE



COMPLINK may be more effecient than SIMPLINK if:

- Special status is relatively stable
- Relatively few special status records
- Heavy batch serial access

Figure 24

Every application is different. Its success depends much more on how well you have made it perform in the eyes of your users than on how well you have applied standard techniques and principles.

Sometimes you do everything "right" and create a performance dog. You may be stuck in the a lose-lose position.

1: Your application has two obviously good linkage paths. Performance is acceptable for all transaction processing.

2. Nightly batch exception reporting is horrendous You can't get your special status reports finished by 8:00 A.M.

3. You establish a linkage allowing direct access by status. Note: You're lucky you didn't blow chain length limits.

4. Your nightly exception reporting is now a piece of cake.

5. You have a new problem. Transaction processing is dying. The extra chain and its volatility are eating resources alive.

6. Now what?

The time has come to revalue your position. Your thought and action process might go like this

1. I can only get batch efficiency using the complexity of an added linkage path.

2. I can only get transaction processing efficiency with a simple structure.

3. Maybe if I combine the two structures I can have the best of both worlds. I'll try a more complex physical structure that I can look at from two logical directions.

4. For my normal records I'll have a detail linked by the normal two linkages. As long as the status isn't really special I'll treat it as just another field.

5. For my special records, those with very special status codes, I'll design another detail linked to status.

6. With rare exceptions, my transaction processing will run well just like it did before.

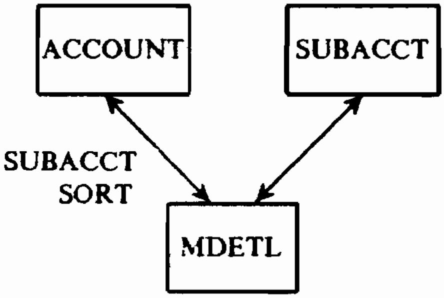
7. My batch exception reports will be printed on time.

8. I think I've got it!

You have abused the data base by making it much more complex. You have been selective in matching your changes to your application. You have been constructive - your application now performs properly.

There are times to follow accepted rules and there are times to write your own rules. The only real problem is knowing when to do which.

USE SORTED CHAINS



SORTED CHAINS MAY BE VERY

USEFUL/EFFICIENT FOR

— "MASTER" detail data

-- non-volatile data

Figure 25

USE SORTED CHAINS (Figure 25)

Sorted chains are powerful tools and carry with them high potential overhead burdens. We should use them carefully but we should not be afraid of them.

Many data base designs could improve functionally if we designed them with sorted chains. We avoid most potential uses because they will perform poorly. Certain cases will perform extremely well.

1. The detail sets are pure linkage sets: they merely serve to define ordered paths between masters.

2. The detail sets are "master" type details. They represent, for example, what might be considered a master record in an indexed file environment.

3. The data in question is stable.

This structure is extremely useful for keeping master implosion and explosion chains. In an accounting application, they can allow us to explode an account number into sorted sub-account to account.

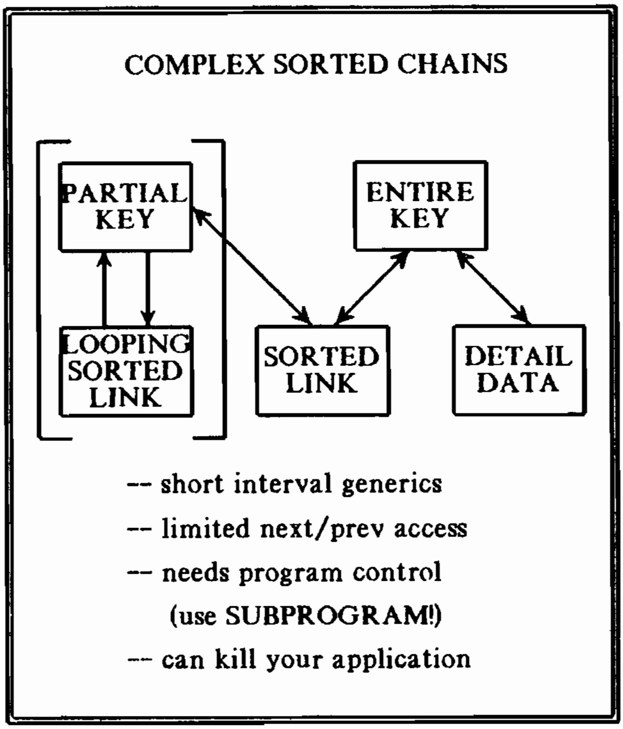
CONSIDER COMPLEX SORT STRUCTURES (Figure 26)

Sorted chains and complex structures are both potential performance bottlenecks. Functional necessity may require their use.

IMAGE is not designed to give logically sequential access of master data sets. Indexed file structures provide logical sequential access but lack most of the capabilities of IMAGE. We're in that potential lose-lose situation again if we need IMAGE capabilities and have to have logical sequential access.

Some programmers solve this problem by maintaining dual data structures. They create the normal IMAGE data base and keep a separate KSAM file to contain key values extracted from their IMAGE master sets. It works well for most of them.

Other programmers attack the problem by setting out to emulate index type structures within IMAGE. I like this approach because it seems more fun to work with and it keeps everything in IMAGE. Once you define the master you follow this basic plan.



COMPLEX SORTED CHAINS

-- short interval generics

-- limited next/prev access

-- needs program control
(use SUBPROGRAM!)

-- can kill your application

Figure 26

1. Define a path to a detail which links your master to your index emulation sets. This detail is optionally sorted by master value on the linkage path coming from the index emulators.

2. Define a detail index set to hold linkages between partial keys. This set links to your interface linkage detail and has two linkages to a detail linkage set.

3. Define a detail index set to hold linkages between partial keys in the master index set. One path is sorted to point to lower level (longer) partial keys. A second optional sorted path allows reversal in sequence.

4. A partial key of null values starts your emulated index.

5. The rest of the index structure contains gradually longer

partial keys. The common increment is two characters.

Index emulation has both good and bad points. You make your own choices since you pay the bills.

1. Short interval generics become possible.

2. Limited next/previous record capabilities are available. If you try to process the data base purely sequentially you will probably regret your decision.

3. This technique requires extensive programmatic control. I suggest a canned subprogram.

4. You may be tempted too much by logical sequential access. Remember that physical access will actually be totally random and expensive.

5. Your application may die.

TASK: REVIEW TP FUNCTION
TRANSFER DELAYS

PLAN: ELIMINATE TRANSFERS
AVOID BUILD/WRECK

GOAL: REDUCE OVERHEAD
REDUCE DELAYS

Figure 27

FUNCTION TO FUNCTION TRANSFER DELAYS (Figure 27)

Batch processing applications tend to isolate functionally similar records into groups (often as physical "batches") and pass them as a unit through a program or program stream. Pure transaction processing applications, on the other hand, provide users the ability to enter virtually any transaction with assurance that processing resources will be available. In theory, no transaction can be predicted before the user presents it to the application.

No matter how well we process a transaction, we are judged harshly if we take too long getting ready to process it. A beautifully performing program gives poor performance if it takes too long to begin executing. Our users will justifiably demand rapid transition between functions.

Transfers from one function to another can also involve considerable overhead. If this overhead is too high the machine can become so bogged down it has little left for its true job, processing transactions. Successful applications spend their resources processing, not getting ready to process.

We usually test programs for their ability to process transactions efficiently. We often forget that much more than efficient programs is needed for effective applications. We have to be concerned with function to function transfers if we expect success.

TASK: Review function-to-function transfers.

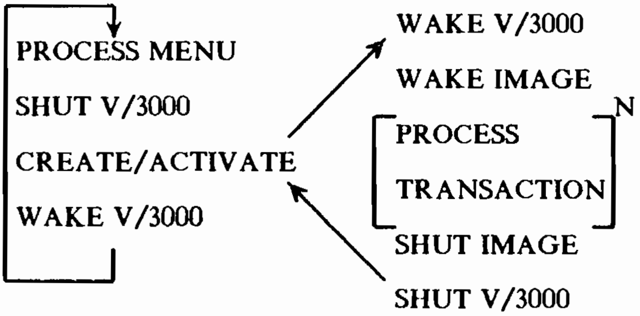
PLAN: Try to eliminate transfers if possible. When transfers cannot be eliminated, try to reduce their cost by reducing their overhead.

GOAL: Reduce system overhead in general. Reduce transfer delays to improve user perception of performance.

MAGNIFICENT MASOCHISM (Figure 28)

Transaction processing applications are often complex. Demands for extreme flexibility stretch programmers skills to the limit. Process handling is an easily implemented

MAGNIFICENT MASOCHISM



PROCESS MENU

SHUT V/3000

CREATE/ACTIVATE

WAKE V/3000

WAKE V/3000

WAKE IMAGE

PROCESS

TRANSACTION

N

SHUT IMAGE

SHUT V/3000

POWERFUL, VALUABLE TOOL

HOWEVER

N = 1 Tremendous overhead

N = few Not terrible

N = many Not really too bad

Figure 28

technique often used to help simplify a complex situation. It can be quite effective in the proper circumstances and can beat the machine senseless when misused.

Most users have plenty to keep them busy. They justifiably require the simplest possible interface between themselves and the machine. Most commercial software packages are menu driven to help win acceptance in the market.

Menu driven applications are easy to design and document. They can also be reasonably simple to program and test. Their biggest potential problem is a tendency toward tedious maintenance and relatively poor performance.

Modular programming techniques reduce the difficulty of maintenance. The most modular technique involves isolating individual or similar functions in separate programs which can be controlled using process handling. A selection of a menu item triggers a programmatic execution of the appropriate program. In effect, the menu program issues predefined "run" commands which are functionally invisible to the user.

Function invisibility is not true invisibility. The user sees the execution of the internal "run" as a delay in processing. There is considerable overhead associated with both starting and stopping a program. The overhead to execute a normal transaction is usually much less than the overhead spent invoking its program.

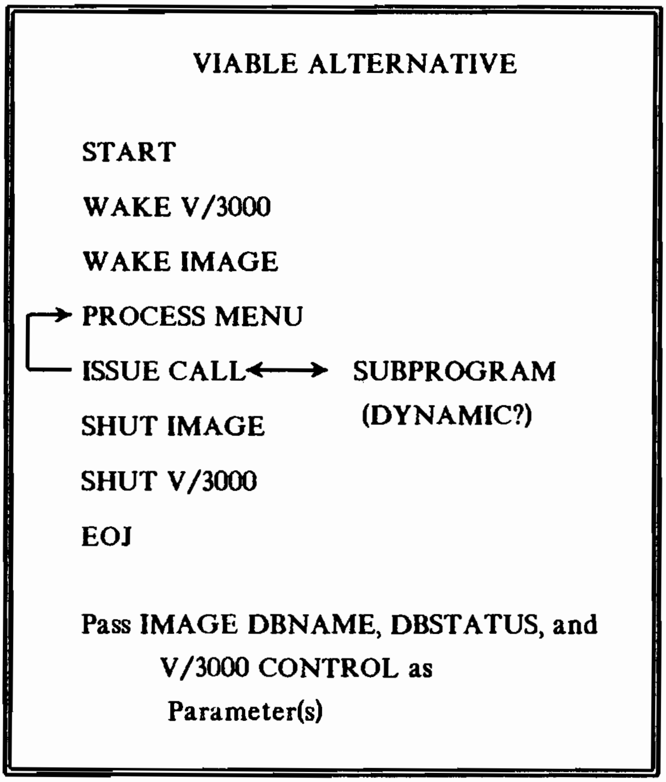
When the selected program will execute numerous transactions before returning to the menu the overhead burden may be quite acceptable. As the number of transactions per execution gets smaller, the overhead becomes objectionable. If the design calls for only 1 transaction per execution the application is potentially terminally ill.

A standard technique to allow flexibility without tremendous overhead is to keep a program alive after it has been invoked. Process handling allows reactivation of a suspended program (process) with minimum overhead and delay. V/3000 processing may create some problems, however, and the limit on the number of processes alive within MPE may curtail your ability to use this technique.

VIABLE ALTERNATIVE (Figure 29)

Some programs promise so much power and flexibility that it is impossible to avoid some performance loss. With intelligent programming we can minimize the loss.

Subprograms allow almost as much flexibility as process handling and usually require less overhead. Critical values such as IMAGE and V/3000 control areas can be passed as parameters from caller to callee. A standard technique involves defining a single data area containing the most



VIABLE ALTERNATIVE

START

WAKE V/3000

WAKE IMAGE

PROCESS MENU

ISSUE CALL ⟷ SUBPROGRAM (DYNAMIC?)

SHUT IMAGE

SHUT V/3000

EOJ

Pass IMAGE DBNAME, DBSTATUS, and V/3000 CONTROL as Parameter(s)

Figure 29

common data and passing it to all subprograms as a single parameter.

Stack sizes may be a limit on the use of subprograms. Dynamic subprograms can help control stack size but must be used carefully to avoid excessive overhead of their own. Excessively large stacks inhibit performance and so can excessive stack expansion, contraction, and initialization.

Dynamic subprograms also limit capabilities of the subprogram. Values kept in local storage is lost when you exit the subprogram. If files are opened they must be closed before you exit. More care is needed when you decide to make an existing subprogram dynamic than when you originally design it to be dynamic.

TASK: REVIEW TP
PROCESSING DELAYS

PLAN: SPLIT PROCESSING
OVERLAP PROCESSING

GOAL: IMPROVE RESPONSE
TO USER

Figure 30

PROCESSING DELAYS (Figure 30)

Processing the actual transaction data is the eventual goal of transaction processing. The time taken for processing ranges from a very small percentage of the total transaction time to a very large percentage. When the percentage is too large we have to try to reduce it to a more acceptable level.

Processing normally breaks down into two phases, data validation and data storage or retrieval. Either or both phases may require extensive disc access with corresponding overhead.

Assuming overhead has already been minimized, there is virtually no way to do anything with the data validation phase. If we expect to reduce processing delays, we can make improvements only in the storage or retrieval phase.

MPE contains easily implemented techniques that allow us to improve performance as perceived by the user. Improvement in perceived performance will not reduce machine overhead and may in fact increase it. In the final analysis, though, increased machine overhead is meaningless so long as the user, the authoritative judge of performance, sees improvement.

TASK: Review processing delays.

PLAN: Evaluate data storage and retrieval delays. When excessive, attempt to use MPE capabilities to reduce or eliminate them.

GOAL: Improve perceived performance as measured by the user.

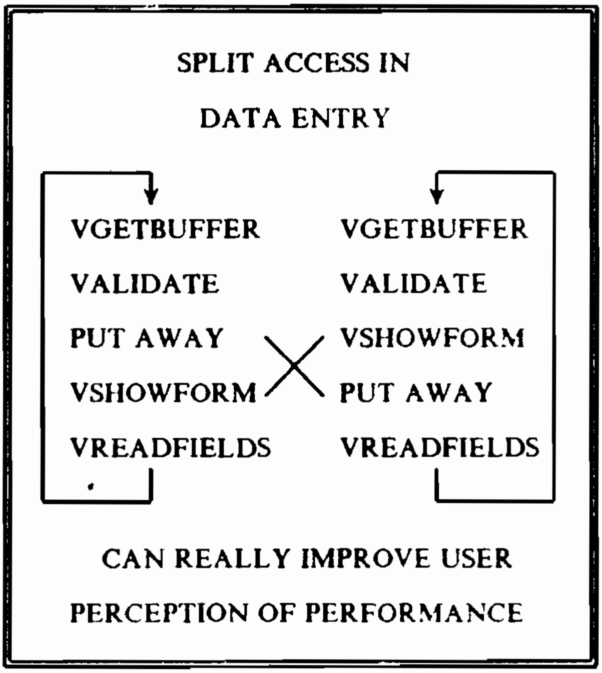

SPLIT ACCESS IN
DATA ENTRY
VGETBUFFER
VALIDATE
PUT AWAY
VSHOWFORM
VREADFIELDS
VGETBUFFER
VALIDATE
VSHOWFORM
PUT AWAY
VREADFIELDS
CAN REALLY IMPROVE USER
PERCEPTION OF PERFORMANCE

Figure 31

SPLIT ACCESS IN DATA ENTRY (Figure 31)

Performance as perceived by the user is or probably should be our primary concern in transaction processing. This is most critical in data entry applications where uniform response helps set up a work rhythm.

Data entry may involve one or many screens per logical transaction. Multi-screen transactions usually benefit from hardware advances such as the ability to use screens downloaded to the terminal. Single screen transaction performance can be influenced by programming.

Screen data is usually validated and then stored on disc. When the storage is in a data base the overhead for storage may take much time. This time is usually the major contributor to delays in processing functions in data entry.

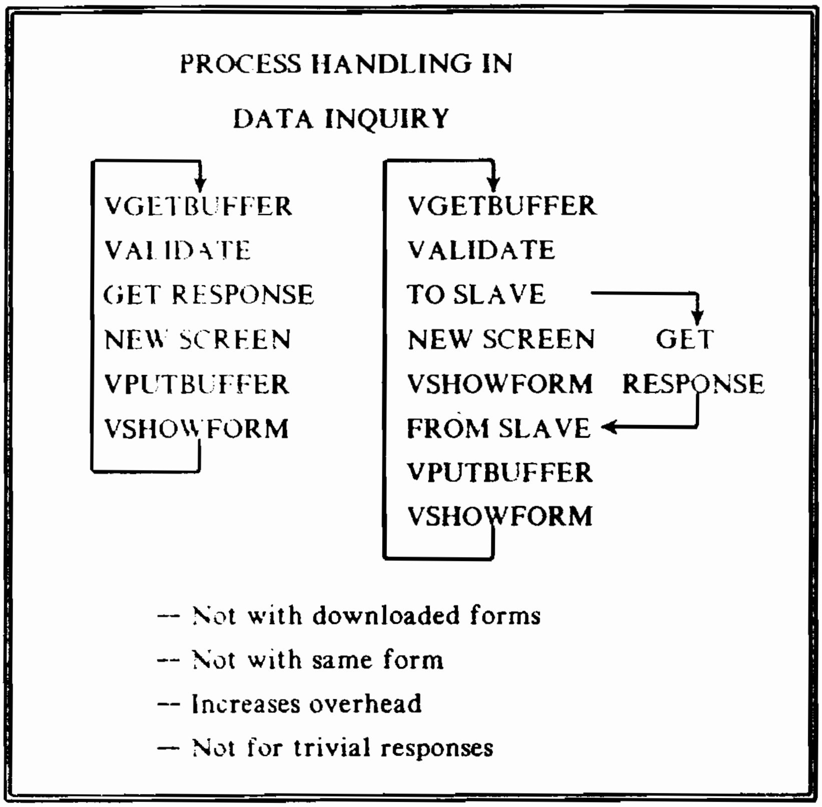
V/3000 can be used to reduce the delays in data entry. Once you have validated the screen and are certain that only a catastrophic failure could keep you from storing the record, you can return the screen to the user for entry of the next record. While the user fills the next screen, you put away your record.

No data will be lost if the record is not stored before the user hits the enter key or a soft key. These attempted transmissions will be ignored until the program issues a read. In all but a few special cases the validated record will be stored well before the user has submitted the next screen.

This technique will not reduce overhead or performance of individual functions within the program. Responsiveness to the user, however, will be improved. The degree of improvement is proportionate to the amount of work being done parallel to the next transaction think time. An additional benefit is that response will become somewhat less dependant upon overall system load.

PROCESS HANDLING IN DATA INQUIRY (Figure 32)

Application programs often have only the simple capability to allow user data inquiry. These programs must perform well but users expect and usually tolerate reasonable response delays. When the data inquiry is part of a larger function those delays must be minimized.



PROCESS HANDLING IN
DATA INQUIRY

VGETBUFFER
VALIDATE
GET RESPONSE
NEW SCREEN
VPUTBUFFER
VSHOWFORM

VGETBUFFER
VALIDATE
TO SLAVE
NEW SCREEN
VSHOWFORM
FROM SLAVE
VPUTBUFFER
VSHOWFORM

GET RESPONSE

-- Not with downloaded forms
-- Not with same form
-- Increases overhead
-- Not for trivial responses

Figure 32

Programs which update existing records have characteristics of both data entry and inquiry. The inquiry part can be a major performance bottleneck. This is quite common since update programs often must retrieve multiple records even though only one may actually be subject to change.

Additional delays are built into such programs if the modifiable response must be displayed on a new screen. User patience is severely tested in many applications because we program serially

1. We read the request and validate its content. We hit the screen fast if we find errors.

2. We are not so friendly with good input. The user must

wait for us to access our data bases to build a reply. Their reward for good input is the chance to watch a cursor blink.

3. We finally respond by sending out a new screen and allow user modification.

4. We satisfy user function but we do it in slow, easy steps.

Process handling can be used to make a program both more responsive and more friendly. We can perform critical time consuming steps in parallel so that delays visible to the program can be invisible to the user.

1. We read requests and handle errors as usual.

2. For good input we immediately send a "please retrieve this data" message to a slave process we have created. The slave was waiting patiently since its only job is to retrieve records from data bases.

3. While the slave gathers records we paint the response screen. This is a friendly act. Our response to good input is probably only a fraction of a second slower than a response to bad input.

4. The faster parallel function waits for the slower to finish. We then retrieve the response buffer from the slave and write it to the screen.

5. Except for a short time needed for process to process communication we have reduced user delay to the delay of the longer parallel process. Even more important, we have become responsive to our users.

This technique is valuable because it improves performance within a technologically limited environment. Changes in technology could make it less valuable or even useless. For example, this technique has no benefit when screens have been downloaded to the terminal since that technology has already eliminated screen painting delays.

We also would not use this technique for trivial responses which take little time to prepare. Process handling absorbs overheads and may not always be cost effective.

TASK: REVIEW V/3000
FORM PROCESSING

PLAN: RESEQUENCE FUNCTIONS
DEFER/DROP FUNCTIONS

GOAL: IMPROVE RESPONSE
REDUCE OVERHEAD

Figure 33

V/3000 FUNCTIONAL SEQUENCING (Figure 33)

Your application defines the V/3000 screens you and your users create for the person to machine interface. Although physical screen design may be critically important to performance it is too broad a subject for this paper. We'll have to assume you've already designed satisfactory screens.

After the screen has been filled, the programmer can begin to manage the processing of the input. Application requirements and V/3000 protocols must be satisfied but the programmer has many options. Some of these options can greatly influence performance.

Application requirements must be met. The programmer should interpret these requirements, however, to see if they can be resequenced more efficiently in the program. Resequencing internal events often changes performance.

V/3000 protocols must also be satisfied. The V/3000 documentation defines a hierarchy among the V/3000 intrinsics and among the functions performed for us by the V/3000 edits. That hierarchy is also open to programmer interpretation. We may be able to improve performance by changing our use of the V/3000 intrinsics.

TASK: Review V/3000 form processing

PLAN: Resequence functions selectively, defer functions when practical, drop functions where possible

GOAL: Improve performance within the program and as perceived by the user.

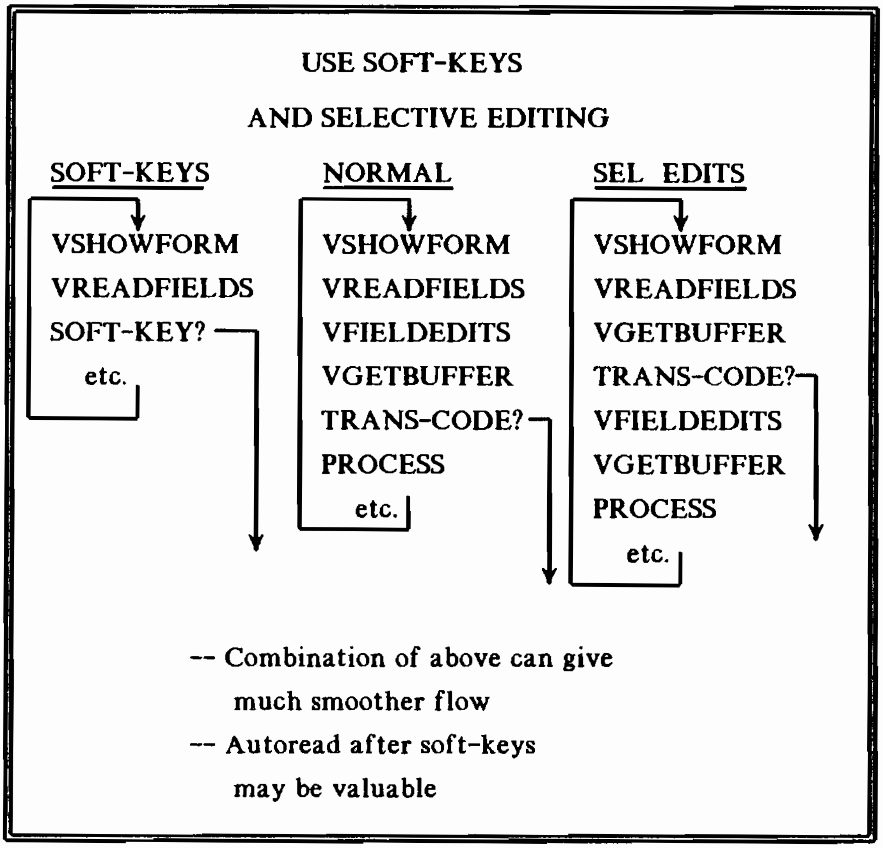
SOFT KEYS AND SELECTIVE EDITING (Figure 34)

Dumb terminals and unsophisticated terminal I-O limit transaction processing. Intelligent terminals and more sophisticated terminal I-O interfaces remove many of these limitations. We may also have to become more intelligent and sophisticated to more fully utilize our better tools.

Soft keys were among the first improvements as terminals began evolving from absolute dumb to somewhat smart. We take them for granted and use them for standard functions such as "exit" or "refresh". Few of us are using them fully. This paper cannot attempt to cover this topic but we can justify looking at a frequently overlooked usage.

Many applications could flow more smoothly if soft keys could be used to trigger processing functions and if screen information could also be available for processing. A common design technique is to require the user to hit the function defining soft key and then hit "enter" to transmit the buffer. This works but it is neither sophisticated nor friendly.

V/3000 can work with the terminal to allow reading the screen after a soft key. We can trigger this function, called auto-read, any time we wish using a simple subprogram or any other



USE SOFT-KEYS

AND SELECTIVE EDITING

SOFT-KEYS
VSHOWFORM
VREADFIELDS
SOFT-KEY?
etc.
NORMAL
VSHOWFORM
VREADFIELDS
VFIELDEDITS
VGETBUFFER
TRANS-CODE?
PROCESS
etc.
SEL EDITS
VSHOWFORM
VREADFIELDS
VGETBUFFER
TRANS-CODE?
VFIELDEDITS
VGETBUFFER
PROCESS
etc.

-- Combination of above can give much smoother flow

-- Autoread after soft-keys may be valuable

Figure 34

technique to set the autoread bit in the V/3000 common area. Autoread can help make an application run more smoothly and be more friendly. That qualifies as a performance improvement.

Another facility we often ignore is the ability to edit data selectively. V/3000 documentation implies that screen edits must preceed program edits. This is a valid standard for most cases but it may not fit comfortably into all situations. Fortunately, the implied sequence is not mandatory.

When we assume that all screen edits must be done before any program edits we may be painting ourselves into a corner.

Why, for example, should we edit an order quantity when the ordered item is an invalid product. A standard workaround is to avoid V/3000 edits in favor of program edits. Like many workarounds, it works but denies us access to a useful subsystem capability.

We can sometimes "have our cake and eat it too" if we become more flexible. Consider the following sequence of events that uses full V/3000 field edits but allows program intervention at a critical point. This is only a simplified example of sequence editing.

1. Read the screen normally using VSHOWFORM and VREADFIELDS
2. Don't edit the screen yet. Issue a call to VGETBUFFER so you can check a critical field in your program
3. If the critical field fails a test, give the screen back with an appropriate message
4. If the critical field is acceptable, go back to "normal" processing. Issue calls to VFIELDEDITS, etc.
5. You have it made. You get program control at a critical stage and you use V/3000 for less critical work.

DEFER OR DROP PROCESSING (Figure 35)

V/3000 gives us much power with little programming effort. When used sensibly, V/3000 can be good for both the programmer and the user. When used only from the programmers point of view, V/3000 can be a nuisance to the user.

V/3000 gives us three standard processing phases. The initialization phase is relatively little used and causes few problems. We seldom use the edit and finish phases absolutely wrong but we often cause excessive overhead and user aggrevation.

Unless there is some special requirements in an application, some edit functions should never be considered in the edit

DEFER/DROP PROCESSING

DROP	AS FOUND	DEFER
IN 7:204 ...	IN 7:204 ...	IN 7:204 ...
FINISH	JUSTIFY RIGHT	FINISH
JUSTIFY RIGHT	FILL LEADING "0"	JUSTIFY RIGHT
		FILL LEADING "0"
ZERO FILL	IT WORKS !!	WHY NOT WAIT?
NOT ALWAYS		AVOID "JIGGLING"
NEEDED		

Figure 35

phase. Whenever we justify or fill a field in an edit phase statement we are performing a potentially wasted function. Even more important, why should we run the risk of having to rewrite a valid field just because we have altered its form? Users object to excessive screen "jiggling".

We have done only part of our job when we isolate functions within logical V/3000 phases. Whenever possible we should avoid the finish phase until we fully accepted the screen. We waste overhead and risk "jiggling" when we perform this work prior to full screen acceptance.

Some functions can be dropped entirely. Until the machine or operating system are changed, we do not have to fill leading blanks with zeros in numeric fields. The machine treats leading blanks as zeros and allows us to drop the fill function if absolutely necessary for performance.

INTRINSICS AND LANGUAGES (Figure 36)

TASK: REVIEW USE OF
INTRINSICS & LANGUAGES

PLAN: UTILIZE GOOD POINTS
AVOID BAD POINTS

GOAL: IMPROVE PERFORMANCE
AVOID NON-PERFORMANCE

Figure 36

Most programmers prefer a particular language and are proficient in it. In the average application the choice of languages will not determine program performance. We cannot guarantee that any one language is inherently better than another in all possible applications.

We also know that the average application often requires specific functions that may not be available in our language of choice. If we work in a multi-lingual shop we may be able to have a special subprogram written in a language that supplies that function. If our shop is single language we either avoid special functions or devise emulation techniques of some sort.

MPE assists all of us by providing intrinsics for many special functions. Many functions not built into a compiler may be available through an intrinsic. This can simplify our programming and make our jobs much easier.

Somehow we will find ways to get our programs running. All too often we immediately have to find a way to get them running more efficiently. Part of that may require a revaluation of how we have used our languages and the intrinsics.

TASK: Reviews use of languages and intrinsics

PLAN: Use strengths of each and try to avoid weaknesses

GOAL: Improve performance. Of utmost importance, avoid non-performance.

INTRINSICS

PROVIDE: CAPABILITIES
CONVENIENCE
UNIFORMITY

NOT ALWAYS: MOST SPEED
MOST EFFICIENCY

FOR SIMILAR UTILITARIAN
FUNCTIONS, COMPILER
ROUTINES USUALLY
BEAT INTRINSICS

Figure 37

INTRINSICS (Figure 37)

Intrinsics are powerful generalized routines provided with MPE. Some allow limited access to highly specialized functions and others give standardized access to common utility functions.

Intrinsics offer the only access to many functions not available unless you program at the machine instruction level and execute in privileged mode. Access to I-O, for example, is only available through the intrinsics. Compiler library modules invoked by the compilers eventually call intrinsics for all file access.

All "special capabilities" are available only through the intrinsics. Process handling and data management are invoked using a relatively small subset of the intrinsics. Programmatic communication between computers is handled in similar fashion.

Intrinsics also provide convenient access to many utilitarian functions. Binary to Ascii and Ascii to Binary conversions are common examples. Serial table searches and character transformations are also available.

Specific parameter format and sequence requirements guarantee uniformity. Except for individual compiler conventions and limitations, once you learn to use an intrinsic in one language you should be able to use it in all languages.

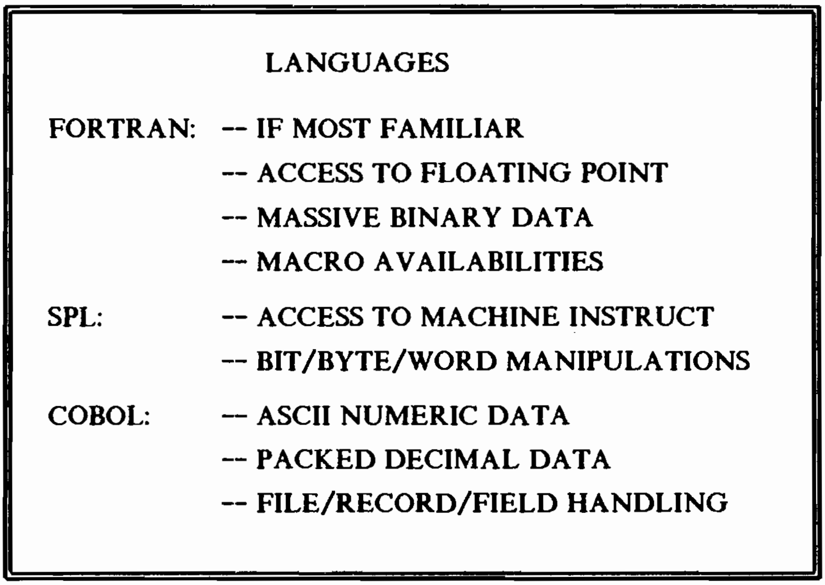
SPL comes closest to accessing intrinsics in strict conformance with their documentation. All other languages provide higher level interfaces to one degree or another.

Intrinsics are carefully coded and function efficiently. They are also highly generalized. This generalization leads to relatively high internal overhead. The intrinsic may expend a good portion of its efforts isolating the particular subfunction it is being asked to perform.

Intrinsics will generally be marginally more efficient for file manipulation than the corresponding compiler modules. The difference is slight and seldom justifies the required attention to minute detail. This generalization is meaningless when we use the intrinsics to reach capabilities not available within a particular language.

Compiler routines usually handle discrete data manipulations more efficiently than would the corresponding intrinsic. Even when the compiler routine internally invokes an intrinsic it will not be meaningfully degraded.

Arithmetic functions are particularly well handled by compiler routines. Many of these routines generate highly efficient machine code. The COBOL compiler uses packed decimal instructions extensively and is surprisingly efficient in handling numeric data.



LANGUAGES

FORTRAN: -- IF MOST FAMILIAR
-- ACCESS TO FLOATING POINT
-- MASSIVE BINARY DATA
-- MACRO AVAILABILITIES

SPL: -- ACCESS TO MACHINE INSTRUCT
-- BIT/BYTE/WORD MANIPULATIONS

COBOL: -- ASCII NUMERIC DATA
-- PACKED DECIMAL DATA
-- FILE/RECORD/FIELD HANDLING

Figure 38

LANGUAGES (Figure 38)

No language is perfect. Each has its special strengths and weaknesses. A language should be evaluated according to how it fits into the application environment.

FORTRAN is an established language with many strong supporters. Programmers either love it or hate it. Either way, there are times when it should probably be your language of choice.

1. If FORTRAN is most familiar to you, why not use it? Most of us usually produce better work when we work with a known quantity.

2. FORTRAN offers excellent high-level access to floating

point arithmetic. This is a major strength of FORTRAN and often virtually necessitates its use. FORTRAN is often used to write subprograms accessible from other languages.

3. FORTRAN works very close to the machine when doing binary arithmetic. Massive calculations with binary data often justify FORTRAN for performance reasons.

4. FORTRAN includes many high-level macro constructs. These may make FORTRAN more useful than other languages.

5. Unfortunately, FORTRAN performs poorly with Ascii numeric data and in its current implementation cannot handle packed decimal numerics. Since these data formats are used extensively in commercial applications, FORTRAN is usually a poor choice there.

SPL is the lowest level language on the HP3000. As such, it is potentially the most efficient. It is also relatively tedious compared to most high-level languages.

1. Although SPL is potentially more efficient than any other languages it will probably not improve performance enough to justify wholesale use in commercial applications. Higher level languages are simply better suited for general purpose use.

2. SPL usually fits in best when used for writing specialized subprograms. Some functions are simply not handled well by high-level languages.

3. Machine instructions can be reached in SPL. This is particularly valuable for string and bit manipulations.

4. SPL is most suited for work with low level data. It works especially well at the word level and below. SPL handles records and fields well but the source code tends to become cumbersome and difficult to maintain in large programs.

COBOL is the most widely used high-level commercial language.

Most programmers hate COBOL, some grudingly accept it, and almost none will publicly admit a preference for it. For all its faults it remains the language of choice in most shops.

1. COBOL has a reputation as an inefficient language. COBOL is definitely not the most efficient language but its wide use implies acceptable performance. There is little question that it is highly effective.

2. COBOL is quite efficient in handling numeric Ascii data. Ascii numeric functions are performed using packed decimal arithmetic. Conversions between Ascii and packed decimal data are done efficiently by native machine instructions.

3. Packed decimal data is a standard data format in COBOL under MPE. The ability to deal with packed numerics is a definite advantage for conversions or interfaces with the best known Brand-X computer.

4. COBOL is a high-level language designed to be effective with high-level data. It is most powerful when used to process files, records within files, and fields within records. This is probably the main reason for its wide acceptance.

SECTION 3 MIXED BAG

NITS (USUALLY)

WHY:
- -- Retest functions in subprograms?
- -- Pass long/short parameter lists?
- -- Nibble at extra data segments?
- -- Use absolute values?
- -- Use abnormal data formats?
- -- Parse in COBOL or FORTRAN?
- -- Initialize tables with loops?
- -- Waste stack flagrantly?
- -- Save most of nothing?

Figure 39

NITS (USUALLY) (Figure 39)

We concern ourselves too much with minor programming considerations. They usually influence performance only slightly. That is not justification, however, for us to code inefficiently.

1. Unless we are writing highly generalized subprograms we probably waste overhead retesting functions. We know why we call a subprogram. In most cases we would be more efficient by writing subprograms with multiple entry points. This avoids retesting to identify our request and also results in more readable programs.

2. Many words have been written about the effect of

parameter lists on subprogram efficiency. Long parameter lists are less efficient than short ones. On the other hand, artifically short lists are eventually less efficient if we waste CPU power loading and unloading common control areas. How we enter a subprogram is probably meaningless unless we are doing very little work within the subprogram.

3. Extra data segements have many uses. They can be used quite efficiently but not without added overhead. Whenever data must be moved to or from an extra data segment we should move as much as practical per access. Moving a single word twice costs about the same as moving over 1000 words once.

4. Absolute numeric values are often needed. They cause extra overhead, however, when used in calculations, especially when the result of the calculation is an absolute value. Absolute values force the compiler to generate extra code to guarantee proper results.

5. Packed decimal data has a natural format containing an odd number of numeric digits, each taking up one 4 bit nibble. These plus a 4 bit numeric sign fills complete bytes. If you specify an even number of decimal digits for packed data you force the compiler to do extra work controlling the low order nibble.

6. Character strings are best parsed by specialized routines which utilize special machine instructions. These routines may be designed directly into a compiler or are easily written in SPL. Complex byte manipulations and loop constructs written in highlevel languages are inefficient and should be avoided in most cases.

7. Tables are often initialized using a program loop which

indexes through the table depositing values along the way. This is relatively inefficient and cumbersome. You can save overhead by filling only the first entry and then performing an overlapping move into the rest of the table.

8. Stack space is valuable and should be used with care. Program-directed literals save the stack area needed for valued data elements. But programmers often value documentation over stack. Even so, using a 132 character data element full of spaces to clear a print record is intolerable flagrant waste.

9. Programmers should be efficient but they have to be reasonable first. There is no way we can justify spending time optimizing a small inefficiency when the same effort could be more productive elsewhere. Inefficient coding in insignificant routines does not make inefficient programs.

DON'T _ BUT ...

DON'T	Believe everything
BUT	Believe something
DON'T	Challenge everything
BUT	Challenge something
DON'T	Optimize everything
BUT	Optimize something
DON'T	Quantify everything
BUT	Quantify something

Figure 40

DON'T ... BUT ... (Figure 40)

What was impossible or ridiculous in the past may be standard practice today. What is absolutely true today will quite often be made false by the technology of tomorrow. There are no absolutes but there may be some valuable guidelines.

I'm a programmer, not a philosopher. In Figure 40, I put some comments that have a certain meaning to me. I think they will be more meaningful to you if you supply your own interpretations and meanings.

Good luck and good programming.

SECTION 4 SAMPLE SOURCE LISTINGS

```
* **********************************************************
* THIS IS THE MASTER PROGRAM USED AS A DRIVER TO SHOW THE
*  TECHNIQUES OF HAVING SORTS EXECUTE IN A SLAVE PROGRAM
* THIS PROGRAM COULD HAVE DRIVEN MULTIPLE SLAVE SORT
*  PROGRAMS AT THE SAME TIME IF I WANTED TO TAKE UP THAT
*  MUCH CODE SPACE IN THE HANDOUT
* **********************************************************

$CONTROL USLINIT
 IDENTIFICATION DIVISION.
* DRIVER TO TEST PROGRAM SORTS1
 PROGRAM-ID. FREPSRSS.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. X.
 OBJECT-COMPUTER. Y.
 SPECIAL-NAMES.
     CONDITION-CODE IS COND-CODE.
 INPUT-OUTPUT SECTION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  UNBLOCKER                 PIC S9999 COMP.
 01  SAVED                     PIC S9999 COMP.
 01  PROG                      PIC X(10) VALUE "SORTS1R".
 01  PIN                       PIC S9999 COMP.
 01  BFACT                     PIC S9999 COMP VALUE 13.
 01  DSEG-IND                  PIC S9999 COMP.
 01  DSEG-ID                   PIC S9999 COMP VALUE 43.
 01  DSEG-LEN                  PIC S9999 COMP VALUE 79.
 01  REC-AREA.
     05  CURR-COUNT            PIC S9999 COMP VALUE 0.
     05  EACH-REC              PIC X(12) OCCURS 13 TIMES.
 01  DATA-REC.
     05  SEND-COUNT            PIC 999 VALUE 55.
     05  SEND-PROC             PIC 9(9).
 PROCEDURE DIVISION.
```

```
START-ITS SECTION.
START-IT.
    CALL INTRINSIC "GETDSEG" USING
        DSEG-IND DSEG-LEN DSEG-ID
    IF COND-CODE < 0
        CALL INTRINSIC "QUIT" USING 101
    .
    CALL INTRINSIC "CREATE" USING
        PROG \\ PIN
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 201
    .
    PERFORM SEND-EMS UNTIL
        SEND-COUNT < 1
    MOVE -1 TO CURR-COUNT
    PERFORM ACTUAL-SENDS
    MOVE 0 TO CURR-COUNT
    PERFORM GET-EMS UNTIL
        CURR-COUNT = -1
    DISPLAY "THAT'S ALL, FOLKS"
    STOP RUN
    .
GET-EMS SECTION.
GET-EM.
    CALL INTRINSIC "DMOVIN" USING
        DSEG-IND 0 79 REC-AREA
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 204

    MOVE 1 TO UNBLOCKER
    PERFORM DOITS UNTIL
        UNBLOCKER > CURR-COUNT
    IF CURR-COUNT NOT = -1
        CALL INTRINSIC "ACTIVATE" USING
            PIN 3
        IF COND-CODE NOT = 0
```

```
           CALL INTRINSIC "QUIT" USING 205

 DOITS SECTION.
 DOIT.
     DISPLAY "FROM SORT " EACH-REC(UNBLOCKER)
     ADD 1 TO UNBLOCKER
     .
 SEND-EMS SECTION.
 SEND-EM.
     MOVE 0 TO CURR-COUNT
     PERFORM LOAD-BUFFERS UNTIL
         CURR-COUNT = BFACT OR SEND-COUNT < 1
     PERFORM ACTUAL-SENDS
     .
 ACTUAL-SENDS SECTION.
 ACTUAL-SEND.
     CALL INTRINSIC "DMOVOUT" USING
         DSEG-IND 0 79 REC-AREA
     IF COND-CODE NOT = 0
         CALL INTRINSIC "QUIT" USING 303
     .
     CALL INTRINSIC "ACTIVATE" USING PIN 3
     IF COND-CODE NOT = 0
         CALL INTRINSIC "QUIT" USING 304
     .
     MOVE 0 TO CURR-COUNT
     .
 LOAD-BUFFERS SECTION.
 LOAD-BUFFER.
     SUBTRACT 1 FROM SEND-COUNT
     CALL INTRINSIC "PROCTIME" GIVING SEND-PROC
* IF YOU RUN THIS TEST VERSION, DON'T ASSUME COBOL IS
*  SLOW JUST BECAUSE THE CPU TIME SHOWS 4-5 MILLISECOND
*   BETWEEN RECORDS.  THAT TIME PRIMARILY REPRESENTS
*    THE CPU TIME NEEDED TO DO THE DISPLAY STATEMENT
     ADD 1 TO CURR-COUNT
     MOVE DATA-REC TO EACH-REC(CURR-COUNT)
     DISPLAY "TO SORT " DATA-REC
     .
* ***************************************************************
* ***************************************************************
* *************************************************************
* THIS IS A SAMPLE OF A SLAVE SORT PROGRAM CONTROLLED BY
*  PROCESS HANDLING TO ALLOW SORTING OF RECORDS OUTSIDE
*  THE MASTER PROGRAM
* MULTIPLE SUCH PROGRAMS CAN BE CONTROLLED AT THE SAME
*  TIME BY THE MASTER
* *************************************************************
$CONTROL USLINIT
 IDENTIFICATION DIVISION.
* KEPT AS SORTS1S, PROGRAM KEPT AS SORTS1R
*  RECEIVES RECORDS FROM SORTMASR
*  SORTS
*  RETURNS SORTED RECORDS TO SORTMASR
 PROGRAM-ID. FREPSRTS.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. X.
 OBJECT-COMPUTER. Y.
 SPECIAL-NAMES.
     CONDITION-CODE IS COND-CODE.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT SORTFILE ASSIGN TO "TEMPSORT,,,,100".
 DATA DIVISION.
 FILE SECTION.
 SD  SORTFILE.
 01  SORTREC.
     05  KEY1                PIC 999.
     05  FILLER              PIC X(7).
     05  KEY2                PIC 99.
 WORKING-STORAGE SECTION.
 01  SORT-FLAG               PIC XX VALUE LOW-VALUES.
 01  TO-WAIT                 PIC S9999 COMP VALUE 3.
 01  BFACT                   PIC S9999 COMP VALUE 13.
 01  DSEG-IND                PIC S9999 COMP.
 01  DSEG-ID                 PIC S9999 COMP VALUE 43.
 01  DSEG-LEN                PIC S9999 COMP VALUE 79.
 01  REC-AREA.
     05  CURR-COUNT          PIC S9999 COMP VALUE 0.
     05  EACH-REC            PIC X(12) OCCURS 13 TIMES.
 PROCEDURE DIVISION.
 START-ITS SECTION.
```

```
START-IT.
    CALL INTRINSIC "GETDSEG" USING
        DSEG-IND DSEG-LEN DSEG-ID
    IF COND-CODE < 0
        CALL INTRINSIC "QUIT" USING 101

    .
    SORT SORTFILE
        ASCENDING KEY KEY2
        DESCENDING KEY KEY1
        INPUT PROCEDURE GET-INS
        OUTPUT PROCEDURE SEND-BACKS
    STOP RUN

    .
SEND-BACKS SECTION.
SEND-BACK.
    MOVE LOW-VALUES TO SORT-FLAG
    MOVE 0 TO CURR-COUNT
    PERFORM GET-AND-SENDS UNTIL
        SORT-FLAG = HIGH-VALUES
    PERFORM SEND-DATAS
    MOVE 0 TO TO-WAIT
    MOVE -1 TO CURR-COUNT
    PERFORM SEND-DATAS

    .
SEND-DATAS SECTION.
SENDING.
    CALL INTRINSIC "DMOVOUT" USING
        DSEG-IND 0 79 REC-AREA
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 102

    .
    CALL INTRINSIC "ACTIVATE" USING 0 TO-WAIT
    IF COND-CODE NOT = 0
        CALL INTRINSIC "QUIT" USING 103

    .
    MOVE 0 TO CURR-COUNT

    .
GET-AND-SENDS SECTION.
GET-AND-SEND.
    IF CURR-COUNT NOT < BFACT
        PERFORM SEND-DATAS

    .
```

```
     RETURN SORTFILE AT END
         MOVE HIGH-VALUES TO SORT-FLAG
     .
     IF SORT-FLAG NOT = HIGH-VALUES
         ADD 1 TO CURR-COUNT
         DISPLAY "FROM SORT IN SORTS1R " SORTREC
         MOVE SORTREC TO EACH-REC(CURR-COUNT)
     .
 GET-INS SECTION.
 GET-IN.
     PERFORM GET-IN-LOOPS UNTIL
         CURR-COUNT = -1
     .
 GET-IN-LOOPS SECTION.
 GET-IN-LOOP.
     CALL INTRINSIC "DMOVIN" USING
         DSEG-IND 0 79 REC-AREA
     IF COND-CODE NOT = 0
         CALL INTRINSIC "QUIT" USING 111
         STOP RUN
     .
     PERFORM UNLOAD-EMS UNTIL
         CURR-COUNT < 1
     IF CURR-COUNT NOT = -1
         CALL INTRINSIC "ACTIVATE" USING 0 TO-WAIT
         IF COND-CODE NOT = 0
             CALL INTRINSIC "QUIT" USING 105
     .
 UNLOAD-EMS SECTION.
 UNLOAD-EM.
     RELEASE SORTREC FROM EACH-REC(CURR-COUNT)
     DISPLAY "TO SORT IN SORTS1R " EACH-REC(CURR-COUNT)
     SUBTRACT 1 FROM CURR-COUNT
     .
* *************************************************************
* *************************************************************


* **********************************************************
* THIS IS A DRIVER FOR A SHORT TEST RUN SHOWING SOME OF
*  THE COMPARISONS BETWEEN COBOL AND FORTRAN PERFORMANCE
* IT PASSES ASCII NUMERICS TO A COBOL SUBPROGRAM AND A
*  FORTRAN SUBPROGRAM
* IT COLLECTS PROCTIMES FOR 1000 CALLS TO THESE SUBPROGRAMS
* THE SUBPROGRAMS EACH ADD 100 6 DIGIT ASCII NUMBERS
*  TO A COUNTER AND THEN RETURN
* THE TIMES ARE REPRESENTATIVE BUT ARE SLIGHTLY LONG
*  SINCE I HAVE NOT BACKED OUT THE TIME FOR THE LOOP
*  CONTROLLING THE 1000 CALLS. THAT TIME IS SMALL
* WHEN THIS WAS TESTED ON A SERIES III IN DECEMBER OF
*  1981, THE RESULTS WERE AS FOLLOWS:
*    COBOL TOOK   16635 CPU MILLISECONDS
*    FORTRAN TOOK 76217 CPU MILLISECONDS
* *********************************************************

$CONTROL SOURCE,USLINIT
 IDENTIFICATION DIVISION.
* KEPT AS FORCOBDA
 PROGRAM-ID. TESTFORD.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  STAMPS                  PIC S9(9) COMP.
 01  STAMPE                  PIC S9(9) COMP.
 01  STAMPD                  PIC 9(6).
 01  THE-SUM                 PIC 9(8).
 01  THE-SUMD                PIC 9(8).
 01  THE-COUNT               PIC 999 VALUE 0.
 01  THE-TABLE.
     05  THE-NUM             PIC 9(6) OCCURS 100 TIMES
         INDEXED BY THE-IND.
 PROCEDURE DIVISION.
 START-OUT.
     DISPLAY "COMPARISON USING 100 6 DIGIT ASCII ENTRIES"
     PERFORM LOAD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     CALL "COBADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM COBOLII, SUM = " THE-SUMD
     CALL "FORADD" USING @THE-TABLE @THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM FORTRAN, SUM = " THE-SUMD
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM COB-SHOT 1000 TIMES
```

```
    CALL INTRINSIC "PROCTIME" GIVING STAMPE
    COMPUTE STAMPD = STAMPE - STAMPS
    DISPLAY "1000 CALLS TO COBOLII SUMMATION SUBPROGRAM " STA
    CALL INTRINSIC "PROCTIME" GIVING STAMPS
    PERFORM FOR-SHOT 1000 TIMES
    CALL INTRINSIC "PROCTIME" GIVING STAMPE
    COMPUTE STAMPD = STAMPE - STAMPS
    DISPLAY "1000 CALLS TO FORTRAN SUMMATION SUBPROGRAM " STA
    STOP RUN
    .
LOAD-EM.
    ADD 5 TO THE-COUNT
    MOVE THE-COUNT TO THE-NUM(THE-IND)
    .
COB-SHOT.
    CALL "COBADD" USING THE-TABLE THE-SUM
    .
FOR-SHOT.
    CALL "FORADD" USING @THE-TABLE @THE-SUM
    .
* ************************************************************
* ************************************************************


* ***********************************************************
* THIS IS THE COBOL SUBPROGRAM MENTIONED IN THE PRECEEDING
*  COBOL MAIN PROGRAM. IT IS AN ASCII NUMBER CRUNCHER
* ***********************************************************

$CONTROL SOURCE,SUBPROGRAM
 IDENTIFICATION DIVISION.
* KEPT AS ADDCOBSA
 PROGRAM-ID. COBADD.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  THE-COUNT                PIC S9(9) COMP-3.
 LINKAGE SECTION.
 01  THE-TABLE.
     05  THE-NUM              PIC 9(6) OCCURS 100 TIMES
         INDEXED BY THE-IND.
 01  THE-SUM                  PIC 9(8).
 PROCEDURE DIVISION USING THE-TABLE THE-SUM.
 START-OUT.
     MOVE 0 TO THE-COUNT
     PERFORM ADD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     MOVE THE-COUNT TO THE-SUM
     GOBACK
     .
 ADD-EM.
     ADD THE-NUM(THE-IND) TO THE-COUNT
     .
* ************************************************************
* ************************************************************


C *********************************************************
C THIS IS THE FORTRAN SUBPROGRAM MENTIONED IN THE
C  PRECEEDING COBOL MAIN PROGRAM
C *********************************************************
C
$CONTROL LIST,MAP,LOCATION,STAT
      SUBROUTINE FORADD(INMAT,SUM)
      CHARACTER*6 INMAT(100)
      CHARACTER*8 SUM
      INTEGER*4 OUTPUT
      OUTPUT = 0
      DO 60 I=1,100
  60  OUTPUT = OUTPUT + JNUM(INMAT(I))
      SUM = STR(OUTPUT,8)
      RETURN
      END

* ************************************************************
* ************************************************************


* *********************************************************
* THIS IS A DRIVER FOR A SHORT TEST RUN SHOWING SOME OF
*  THE COMPARISONS BETWEEN COBOL AND FORTRAN PERFORMANCE
* IT PASSES BINARY DOUBLE WORDS TO A COBOL SUBPROGRAM
*  AND A FORTRAN SUBPROGRAM
* IT COLLECTS PROCTIMES FOR 1000 CALLS TO THESE SUBPROGRAMS
* THE SUBPROGRAMS EACH ADD 100 BINARY DOUBLE WORDS
*  TO A COUNTER AND THEN RETURN
* THE TIMES ARE REPRESENTATIVE BUT ARE SLIGHTLY LONG
*  SINCE I HAVE NOT BACKED OUT THE TIME FOR THE LOOP
*  CONTROLLING THE 1000 CALLS. THAT TIME IS SMALL
* WHEN THIS WAS TESTED ON A SERIES III IN DECEMBER OF
*  1981, THE RESULTS WERE AS FOLLOWS:
*    COBOL TOOK   9942 CPU MILLISECONDS
*    FORTRAN TOOK 2499 CPU MILLISECONDS
* **********************************************************
$CONTROL SOURCE,USLINIT
 IDENTIFICATION DIVISION.
* KEPT AS TESTFORD
 PROGRAM-ID. FORCOBDB.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  STAMPS                  PIC S9(9) COMP.
 01  STAMPE                  PIC S9(9) COMP.
 01  STAMPD                  PIC 9(6).
 01  THE-SUM                 PIC S9(9) COMP.
 01  THE-SUMD                PIC 9(8).
 01  THE-COUNT               PIC 999 VALUE 0.
 01  THE-TABLE.
     05  THE-NUM             PIC S9(9) COMP OCCURS 100 TIMES
         INDEXED BY THE-IND.
 PROCEDURE DIVISION.
 START-OUT.
     DISPLAY "COMPARISON USING 100 DOUBLE ENTRIES"
     PERFORM LOAD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     CALL "COBADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM COBOLII, SUM = " THE-SUMD
     CALL "FORADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM FORTRAN, SUM = " THE-SUMD
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM COB-SHOT 1000 TIMES
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
     DISPLAY "1000 CALLS TO COBOLII SUMMATION SUBPROGRAM " STA
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM FOR-SHOT 1000 TIMES
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
     DISPLAY "1000 CALLS TO FORTRAN SUMMATION SUBPROGRAM " STA
     STOP RUN
     .
 LOAD-EM.
     ADD 5 TO THE-COUNT
     MOVE THE-COUNT TO THE-NUM(THE-IND)
     .
 COB-SHOT.
     CALL "COBADD" USING THE-TABLE THE-SUM
     .
 FOR-SHOT.
     CALL "FORADD" USING THE-TABLE THE-SUM
     .
* ************************************************************
* ************************************************************


* *********************************************************
* THIS IS THE COBOL SUBPROGRAM MENTIONED IN THE
*  PRECEEDING COBOL MAIN PROGRAM
* *********************************************************

$CONTROL SOURCE,SUBPROGRAM
 IDENTIFICATION DIVISION.
* KEPT AS ADDCOBSB
 PROGRAM-ID. COBADD.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  THE-COUNT                   PIC S9(9) COMP.
 LINKAGE SECTION.
 01  THE-TABLE.
     05  THE-NUM                 PIC S9(9) COMP OCCURS 100 TIMES
         INDEXED BY THE-IND.
 01  THE-SUM                     PIC S9(9) COMP.
 PROCEDURE DIVISION USING THE-TABLE THE-SUM.
 START-OUT.
     MOVE 0 TO THE-COUNT
     PERFORM ADD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     MOVE THE-COUNT TO THE-SUM
     GOBACK

 ADD-EM.
     ADD THE-NUM(THE-IND) TO THE-COUNT

* ************************************************************
* ************************************************************


C *********************************************************
C THIS IS THE FORTRAN SUBPROGRAM MENTIONED
C  IN THE PRECEEDING COBOL MAIN PROGRAM
C *********************************************************
C
$CONTROL LIST,MAP,LOCATION,STAT
       SUBROUTINE FORADD(INMAT,SUM)
       INTEGER*4 INMAT(100)
       INTEGER*4 SUM
       INTEGER*4 OUTPUT
       OUTPUT = 0
       DO 60 I=1,100
  60   OUTPUT = OUTPUT + (INMAT(I))
       SUM = (OUTPUT)
       RETURN
       END

* ************************************************************
* ************************************************************


* *********************************************************
* THIS IS A DRIVER FOR A SHORT TEST RUN SHOWING SOME OF
*  THE COMPARISONS BETWEEN COBOL AND FORTRAN PERFORMANCE
* IT PASSES BINARY DOUBLE WORDS TO A COBOL SUBPROGRAM
*  AND A FORTRAN SUBPROGRAM
* IT COLLECTS PROCTIMES FOR 1000 CALLS TO THESE SUBPROGRAMS
* THE SUBPROGRAMS EACH ADD 100 BINARY DOUBLE WORDS
*  TO A COUNTER AND THEN RETURN
* THE TIMES ARE REPRESENTATIVE BUT ARE SLIGHTLY LONG
*  SINCE I HAVE NOT BACKED OUT THE TIME FOR THE LOOP
*  CONTROLLING THE 1000 CALLS. THAT TIME IS SMALL
* THE FORTRAN SUBPROGRAM WAS THE SAME AS THE ONE USED
*  IN THE PRECEEDING TEST
* I CHANGED THE COBOL SUBPROGRAM TO SHOW HOW COBOL
*  COULD BE SPEEDED UP USING DIFFERENT CODING
* WHEN THIS WAS TESTED ON A SERIES III IN DECEMBER OF
*  1981, THE RESULTS WERE AS FOLLOWS:
*    COBOL TOOK   6527 CPU MILLISECONDS
*    FORTRAN TOOK 2482 CPU MILLISECONDS
* ***********************************************************
$CONTROL SOURCE,USLINIT
 IDENTIFICATION DIVISION.
* KEPT AS TESTFORD
 PROGRAM-ID. FORCOBDB.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  STAMPS                   PIC S9(9) COMP.
 01  STAMPE                   PIC S9(9) COMP.
 01  STAMPD                   PIC 9(6).
 01  THE-SUM                  PIC S9(9) COMP.
 01  THE-SUMD                 PIC 9(8).
 01  THE-COUNT                PIC 999 VALUE 0.
 01  THE-TABLE.
     05  THE-NUM              PIC S9(9) COMP OCCURS 100 TIMES
         INDEXED BY THE-IND.
 PROCEDURE DIVISION.
 START-OUT.
     DISPLAY "COMPARISON USING 100 DOUBLE ENTRIES"
     PERFORM LOAD-EM VARYING THE-IND FROM 1 BY 1
         UNTIL THE-IND > 100
     CALL "COBADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM COBOLII, SUM = " THE-SUMD
     CALL "FORADD" USING THE-TABLE THE-SUM
     MOVE THE-SUM TO THE-SUMD
     DISPLAY "FROM FORTRAN, SUM = " THE-SUMD
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM COB-SHOT 1000 TIMES
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
     DISPLAY "1000 CALLS TO COBOLII SUMMATION SUBPROGRAM " STA
     CALL INTRINSIC "PROCTIME" GIVING STAMPS
     PERFORM FOR-SHOT 1000 TIMES
     CALL INTRINSIC "PROCTIME" GIVING STAMPE
     COMPUTE STAMPD = STAMPE - STAMPS
     DISPLAY "1000 CALLS TO FORTRAN SUMMATION SUBPROGRAM " STA
     STOP RUN
     .
 LOAD-EM.
     ADD 5 TO THE-COUNT
     MOVE THE-COUNT TO THE-NUM(THE-IND)
     .
 COB-SHOT.
     CALL "COBADD" USING THE-TABLE THE-SUM
     .
 FOR-SHOT.
     CALL "FORADD" USING THE-TABLE THE-SUM
     .
* *************************************************************
* *************************************************************


* **********************************************************
* THIS IS THE IMPROVED COBOL SUBROUTINE MENTIONED IN THE
*  PREVIOUS COBOL MAIN PROGRAM.  IT SHOWS THAT A PROGRAM
*  CODED LOOP CONTROL CAN BE MORE EFFICIENT THAN A
*  COMPILER CONTROLLED LOOP IN SOME CASES
* UNLESS HARD PRESSED FOR PERFORMANCE, I WOULD NOT
*  USUALLY PREFER MY OWN LOOP CONTROL
* **********************************************************

$CONTROL SOURCE,SUBPROGRAM
 IDENTIFICATION DIVISION.
* KEPT AS ADDCOBXB
 PROGRAM-ID. COBADD.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  THE-COUNT                PIC S9(9) COMP.
 01  THE-IND                  PIC S9999 COMP.
 LINKAGE SECTION.
 01  THE-TABLE.
```

```
     05  THE-NUM                PIC S9(9) COMP OCCURS 100 TIMES.
 01  THE-SUM                    PIC S9(9) COMP.
 PROCEDURE DIVISION USING THE-TABLE THE-SUM.
 START-OUT.
     MOVE 0 TO THE-COUNT
     MOVE 1 TO THE-IND
     .
 ADD-EM.
     ADD THE-NUM(THE-IND) TO THE-COUNT
     ADD 1 TO THE-IND
     IF THE-IND < 101
         GO TO ADD-EM
     .
     MOVE THE-COUNT TO THE-SUM
     GOBACK
     .
* ***************************************************************
* ***************************************************************
```

```
<< THIS SUBPROGRAM ALLOWS BINARY SEARCH OF PB-RELATIVE >>
<<  CODE. IT REQUIRES A FIXED LENGTH ARGUMENT AND      >>
<<  SENDS BACK A FIXED LENGTH RESULT.                  >>

$CONTROL SUBPROGRAM,SEGMENT=FSEARCH
<< KEPT AS SEARCHFS >>
<< FIXED LEN ARG (WORDS) AND FIXED LEN RESULT (WORDS) >>
BEGIN PROCEDURE FINDFIXED(FOUND,ARG,RESULT);
INTEGER FOUND;
INTEGER ARRAY ARG,RESULT;
BEGIN
EQUATE ARG'WLEN = 2;
EQUATE RESULT'WLEN = 17;
EQUATE STEP'SIZE = ARG'WLEN + RESULT'WLEN;
EQUATE ARG'BLEN = ARG'WLEN * 2;
EQUATE RESULT'BLEN = RESULT'WLEN * 2;
EQUATE NUM'COMPARES = 5;
<< FOR NUM'COMPARES:                                 >>
<< IF TOO LARGE, WASTED TIME; IF TOO SMALL, NO-HITS  >>
<< GENERAL GUIDELINE:                                >>
<< IF NUM'COMPARES ** 2 IS LESS THAN THE NUMBER      >>
<<  OF ARGUMENTS TO BE SEARCHED, THE SEARCH WILL     >>
<<  NOT SUCCEED IN ALL CASES                                 >>
<< IF (NUM"COMPARES -1) ** 2 IS GREATER THAN THE            >>
<<  NUMBER OF ARGUMENTS TO BE SEARCHED, THERE WILL          >>
<<  BE SOME WASTED COMPUTER CYCLES                          >>
INTEGER POS,DISP;
BYTE POINTER RESULTB,BARG;
TOS:=@ARG;
ASSEMBLE (LSL 1);
@BARG:=TOS;
FOUND:="NN";
TOS:=@RESULT;
ASSEMBLE (LSL 1);
@RESULTB:=TOS;
TOS:=1;
ASSEMBLE (LSL NUM'COMPARES);
DISP:=TOS;
POS:=@START'DATA + ((DISP - 1) * STEP'SIZE);
WHILE DISP <> 0 DO
  BEGIN
    DISP:=DISP/2;
    IF POS >= @END'DATA THEN
      POS:=POS - (DISP * STEP'SIZE)
      ELSE
      BEGIN
        TOS:=@BARG;
        TOS:=POS;
        ASSEMBLE (LSL 1);
        TOS:=ARG'BLEN;
        ASSEMBLE (CMPB PB);
        IF = THEN
          BEGIN
            DISP:=0;
            FOUND:="YY";
            TOS:=@RESULT;
            ASSEMBLE (LSL 1);
            TOS:=POS+ARG'WLEN;
            ASSEMBLE (LSL 1);
            TOS:=RESULT'BLEN;
            ASSEMBLE (MVB PB);
          END
          ELSE
          IF < THEN POS:=POS-(DISP * STEP'SIZE)
```

```
            ELSE POS:=POS+(DISP * STEP'SIZE);
      END;
  END;
ASSEMBLE (EXIT 3);
START'DATA:
<< START CONS >>
ASSEMBLE (CON "AB  RESAB                             ");
ASSEMBLE (CON "BB  RESULT     BB                     ");
ASSEMBLE (CON "CC  RESULT    CC               !!     ");
ASSEMBLE (CON "CEF RES CEF                           ");
ASSEMBLE (CON "CEFARESULT CEFA                     **");
ASSEMBLE (CON "DE  RDE                               ");
ASSEMBLE (CON "DEF RDEF                              ");
ASSEMBLE (CON "GHIJRESULT FOR GHIJ----               ");
ASSEMBLE (CON "GHK RESULT FOR GHK..................  ");
ASSEMBLE (CON "KKL RESULT FOR KKL                    ");
END'DATA:
END;
END.

* ************************************************************
* ************************************************************


<< THIS SUBPROGRAM ALLOWS BINARY SEARCH OF PB-RELATIVE >>
<<  CODE. IT REQUIRES A FIXED LENGTH ARGUMENT AND      >>
<<  SENDS BACK A FIXED LENGTH RESULT.                  >>
<< THE RESULT IS STORED IN CODE AS A VARIABLE LENGTH   >>
<<  ENTITY AND THE SUBPROGRAM NEED NOT WASTE SPACE     >>
<<  WITH TRAILING BLANKS.  THIS CAN BE SIGNIFICANT.    >>

$CONTROL SUBPROGRAM,SEGMENT=VSEARCH
<< KEPT AS SEARCHVS >>
<< FIXED LEN ARG (WORDS) AND VARIABLE LEN RESULT (WORDS) >>
BEGIN PROCEDURE FINDVARIABLE(FOUND,ARG,RESULT);
INTEGER FOUND;
INTEGER ARRAY ARG,RESULT;
BEGIN
EQUATE ARG'WLEN = 2;
EQUATE RESULT'WLEN = 1;
EQUATE STEP'SIZE = ARG'WLEN + RESULT'WLEN;
EQUATE ARG'BLEN = ARG'WLEN * 2;
EQUATE RESULT'BLEN = RESULT'WLEN * 2;
EQUATE NUM'COMPARES = 5;
<< FOR NUM'COMPARES:                                          >>
<< IF TOO LARGE, WASTED TIME; IF TOO SMALL, NO-HITS           >>
<< GENERAL GUIDELINE:                                         >>
<< IF NUM'COMPARES ** 2 IS LESS THAN THE NUMBER               >>
<<  OF ARGUMENTS TO BE SEARCHED, THE SEARCH WILL              >>
<<  NOT SUCCEED IN ALL CASES                                  >>
<< IF (NUM"COMPARES -1) ** 2 IS GREATER THAN THE              >>
<<  NUMBER OF ARGUMENTS TO BE SEARCHED, THERE WILL            >>
<<  BE SOME WASTED COMPUTER CYCLES                            >>
INTEGER DSTART,DEND,POS,DISP;
BYTE POINTER RESULTB,BARG;
TOS:=@ARG;
ASSEMBLE (LSL 1);
@BARG:=TOS;
FOUND:="NN";
TOS:=@RESULT;
ASSEMBLE (LSL 1);
@RESULTB:=TOS;
TOS:=1;
ASSEMBLE (LSL NUM'COMPARES);
DISP:=TOS;
POS:=@START'DATA + ((DISP - 1) * STEP'SIZE);
WHILE DISP <> 0 DO
  BEGIN
    DISP:=DISP/2;
    IF POS >= @END'DATA THEN
      POS:=POS - (DISP * STEP'SIZE)
      ELSE
      BEGIN
        TOS:=@BARG;
        TOS:=POS;
        ASSEMBLE (LSL 1);
        TOS:=ARG'BLEN;
        ASSEMBLE (CMPB PB);
        IF = THEN
          BEGIN
            DISP:=0;
            FOUND:="YY";
            TOS:=@RESULT;
            ASSEMBLE (LSL 1);
            TOS:=@DSTART;
            ASSEMBLE (LSL 1);
            TOS:=POS+ARG'WLEN;
            ASSEMBLE (LSL 1);
            TOS:=2;
            ASSEMBLE (MVB PB);
            TOS:=@DEND;
            ASSEMBLE (LSL 1);
            TOS:=POS+ARG'WLEN*2+1;
            ASSEMBLE (LSL 1);
            TOS:=2;
            ASSEMBLE(MVB PB);
            TOS:=@END'DATA+DSTART;
            ASSEMBLE (LSL 1);
            TOS:=2*(DEND-DSTART);
            ASSEMBLE (MVB PB);
          END
          ELSE
          IF < THEN POS:=POS-(DISP * STEP'SIZE)
            ELSE POS:=POS+(DISP * STEP'SIZE);
      END;
  END.
ASSEMBLE (EXIT 3);
START'DATA:
<< START CONS >>
ASSEMBLE (CON "AB  ",0000);     << NO PREVIOUS LEN   >>
ASSEMBLE (CON "BB  ",0003);     << PREVIOUS LEN 03W >>
ASSEMBLE (CON "CC  ",0010);     << PREVIOUS LEN 07W >>
ASSEMBLE (CON "CEF ",0025);     << PREVIOUS LEN 15W >>
ASSEMBLE (CON "CEFA",0029);     << PREVIOUS LEN 04W >>
ASSEMBLE (CON "DE  ",0046);     << PREVIOUS LEN 17W >>
ASSEMBLE (CON "DEF ",0048);     << PREVIOUS LEN 02W >>
ASSEMBLE (CON "GHIJ",0050);     << PREVIOUS LEN 02W >>
ASSEMBLE (CON "GHK ",0060);     << PREVIOUS LEN 10W >>
ASSEMBLE (CON "KKL ",0076);     << PREVIOUS LEN 16W >>
ASSEMBLE (CON -1,-1 ,0083);     << PREVIOUS LEN 07W >>
END'DATA:
ASSEMBLE (CON "RESAB "                            ); << 03W >>
ASSEMBLE (CON "RESULT     BB "                    ); << 07W >>
ASSEMBLE (CON "RESULT    CC               !! "    ); << 15W >>
ASSEMBLE (CON "RES CEF "                          ); << 04W >>
ASSEMBLE (CON "RESULT CEFA                     **"); << 17W >>
ASSEMBLE (CON "RDE "                              ); << 02W >>
ASSEMBLE (CON "RDEF"                              ); << 02W >>
ASSEMBLE (CON "RESULT FOR GHIJ---- "              ); << 10W >>
ASSEMBLE (CON "RESULT FOR GHK.................."  ); << 16W >>
ASSEMBLE (CON "RESULT FOR KKL"                    ); << 07W >>
END;
END.

* ************************************************************
* ************************************************************


<< THIS ROUTINE PERFORMS AN AUTOREAD AGAINST A V/3000 >>
<<  FORM. IT IS CALLED WHENEVER YOU WISH TO HAVE THE   >>
<<  SCREEN READ AFTER THE USER HAS HIT A SOFT-KEY AND >>
<<  YOU DO NOT WISH TO FORCE HIM/HER TO HIT "ENTER"    >>
<<  TO TRIGGER THE READ.                               >>
<< IT IS CALLED JUST AS IF IT WERE "VREADFIELDS"       >>
<<  EXCEPT THAT THE NAME WOULD BE CHANGED              >>
<<                                                     >>
<<     CALL "IMMVREADFIELDS" USING VCONT-AREA          >>
<<                                                     >>
<<                                                     >>
$CONTROL SUBPROGRAM
<< KEPT AS IMMREAD >>
BEGIN PROCEDURE IMMVREADFIELDS(CONT);
INTEGER ARRAY CONT;
BEGIN
PROCEDURE VREADFIELDS(C);
INTEGER ARRAY C;
OPTION EXTERNAL;
CONT(55).(13:2):=%1;
VREADFIELDS(CONT);
CONT(55).(13:2):=0;
END;
END.

* 06*******************************************************03
* ***************************82******************************
```

Improving Database Application Performance Without Changing Your Programs

So your IMAGE Database application is performing poorly? And you need to do something but no single program is the bottleneck? And you have 300 separate programs but only two programmers? Well! this paper is for you! These techniques have been employed on several independent systems and have brought relief and even a cure for poor system performance.

The first step is to determine if your database(s) are truly your performance bottleneck. My favorite tool for this is FILERPT, a program which examines MPE Log File Close records and produces several reports to focus on which files are actually the cause of your disc I/O accesses. If FILERPT does not indicate a significant number of database accesses, it may still serve to assist in improving your system. The version I use serves to separate session and job accesses as well as to sort the accesses in several different ways. The first page of a Sample report is seen in Figure 1.

FILERPT is easy to use. Just turn on MPE File Close Logging, let your system run for a few minutes, hours, days, or maybe a week, and feed the Log files into FILERPT. Your first task will be to CREATE a summary file for the reports to use. The file may reflect the File Closes for the System, Sessions, Jobs, or All accesses. I generally produce all four summary files but the All summary file is a good start. Next, the reports on the summary files are produced. The options available are to sort the report by Logical Records Processed, Physical Blocks Processed, File Close Count, Alphabetical by Account/ Group/ Filename, and Alphabetical by Filename/ Group/ Account. Any percentage of the report may be printed (specify LP before producing the report) but I use 100 percent and 10 percent most frequently.

The reports tell you many things, for example:

1. Which files cause the most disc accesses.
2. What programs should be allocated (most frequently closed).
3. Is someone opening files and not using them?
4. Do the most frequently used files have good blocking factors? Are they having an effect?
5. Are people using multiple copies of popular programs?
6. Are the Database accesses a significant portion of the total disc load?

Once we know the impact of our Database accesses, we need to know the nature of the order of the accesses. Are we doing Serial, Chained, or Directed accesses? Do users conflict with each other? Is Image itself doing the accesses because of other factors?

```
BLOCKS PROCESSED    USAGE REPORT V2.01.01  ON FILE L3148491.DOCUMENT.SYS
TUE, MAY 10, 1983,  3:16 PM
FILE NAME                   TYPE  LDEV    REC COUNT   BLK COUNT        FCLOSE
                                                                        COUNT

  GOOD    .SYSOP   .SYS       0     1*     203,958.     30,076.            6.
          .PUB     .IMACS     0     1*      24,671.     24,285.           74.
  OBPRD11 .DATABASE.VALLIANT  0     1       13,685.     13,685.            1.
          .DOC     .MAINLIB   0     3*      13,213.     13,213.          670.
  TEMPCODE.SMBPUB  .BUTLER    0     3*      12,083.     11,692.           18.
  TEMPLOG .DATABASE.VALLIANT  0     3      299,292.      9,912.            1.
  VLOG0066.PUB     .VALLIANT  0     3      280,960.      8,780.            2.
  SL      .PUB     .SYS       0     1        8,643.      8,643.            3.
  SORTSCR .PUB     .SYS       0     3*       7,453.      7,453.            8.
          .PUB     .MONTREAL  0     1*      43,684.      6,968.          252.
          .PUB     .SUPPORT   0     3        6,091.      6,091.            1.
  TEMPLIST.SMBPUB  .BUTLER    0     3*       5,640.      5,599.           18.
          .SMBPUB  .BUTLER    0     3*      12,850.      4,967.          159.
  PCODE   .SMBSPL  .BUTLER    0     1*      77,172.      4,826.           25.
  OBPRD23 .DATABASE.VALLIANT  0     1        3,730.      3,730.            1.
  IPDIC   .PUB     .SYS       0     1        3,661.      3,661.           27.
          .SHIP    .COLDEV    0     1*      21,144.      3,081.           38.
  OPT     .JOB     .SYS       0     1        2,808.      2,808.            4.
  EDIT2   .DOC     .MONTREAL  0     1        1,735.      1,737.            1.
  INFCNV  .RAPID03 .IMACS     0     1        1,494.      1,494.            1.
  PCODE   .SMBOBJ  .BUTLER    0     1*       1,463.      1,463.           29.
  GOOD    .PUB     .SYS       0     3*       5,825.      1,447.           13.
  STORE   .PUB     .SYS       0     1        1,354.      1,354.           46.
  OBPRD03 .DATABASE.VALLIANT  0     1        1,349.      1,349.            1.
  LOADMAP .PUB     .SYS       0     1        2,500.      1,310.           39.
  TRANSACT.RAPID03 .IMACS     0     1        1,308.      1,308.            1.
  DICTDBM .RAPID01 .IMACS     0     1        1,264.      1,264.            1.
  INFORM  .RAPID02 .IMACS     0     1        1,264.      1,264.            1.
          .INFOBASE.MAINLIB   0     3*       1,192.      1,192.          177.
  SPL     .PUB     .SYS       0     1        1,098.      1,098.           24.
  IPDIC   .RAPID01 .IMACS     0     1        1,042.      1,042.            1.
  TDPCATLG.TDPDATA .HPOFFICE  0     1       15,584.        974.           71.
  TRANSACT.PUB     .SYS       0     1          963.        963.           14.
  TRANCOMP.RAPID01 .IMACS     0     1          846.        846.            1.
  TRANCOMP.RAPID02 .IMACS     0     1          846.        846.            1.
  CATALOG .DOCUMENT.QMIT      0     1       12,467.        822.            4.
  SYSDUMP .PUB     .SYS       0     1          805.        805.           15.
  $OLDPASS.SMBPUB  .BUTLER    0     1          305.        799.            6.
  INFCNV  .PUB     .SYS       0     1          747.        747.            1.
  K#######.PUB     .SYS       0     3*      26,093.        674.           13.
```

Figure 1.

Sample FILERPT Output

Several tools are available to assist in this investigation. First is SURVEYOR which is a poor man's OPT. This program shows when disc I/O's are taking place and who is doing them. With the assistance of SOO or some other tool, we may determine the pin numbers of the users, the files involved, and perhaps the degree of conflict for the disc. Perhaps the best tool for examining disc accesses is IOSTAT4. This tool can show us the actual order, location, user, length, and frequency of disc accesses. Figure 2 gives a sample of the output from IOSTAT4.

The definitive tool for IMAGE Database analysis is DBLOADNG. This tool has been enhanced by me so that it now provides even more hard information about the database content than previous versions. The most critical data it produces is the Elongation of the chains in the datasets. This Elongation is a measure of the unnecessary disc accesses which your application may be experiencing. A related count is the number of External Pointers within a dataset. This term was developed by me to reference the chain pointers which point to the next chain entry outside of the current block. These External Pointers always cause another disc access during the chasing of a chain unless IMAGE already has the data in its buffers. The reduction of the number of External Pointers is often the answer to poor overall database performance.

So, we run DBLOADNG (version 2.7 or later) and sure enough we have External Pointers everywhere! (If you don't, there is still hope but it will require more work.)

The following actions should serve to reduce External Pointers.

1. Modify the Master dataset capacities. A target of 60 to 70% full is usually good. Note that a very small change in capacity can have a very large effect on inefficient pointers.

2. Make the most frequently accessed path (via FILERPT) the primary path if the average chain length is greater than the current primary path.

3. Unload and load the datasets to straighten out the chains.
   a. On Details, the primary path will be rebuilt to have a minimum number of External Pointers (equal to the number of Contiguous Pointers).
   b. On Masters, if there exist Empty Holes these will be filled in by External Pointers to become Internal Pointers.

4. Increase the Blocking Factor of the datasets so that more records get into a block. This will be discussed in more detail later in the paper.

```
FIN  LDEV          XLOG   FN      REL      WHERE      QMISC  PARM1  PARM2  STAT
 34   1   1        507W  READ     SEG   407 011666  000000 000002 153120 001 322.
 59   2   2       1730B  READ     +DB   200 020736  000000 000003 112172 001 407.
  :45:56        32
 87   1  *DIRC     384W  READ     SEG    24 000311  000000 000000 000563 001   0.
 34   2   2        499W  READ     SEG   407 002632  000000 000002 114073 001 295.
 87   1  +DIRC     384W  READ     SEG    24 001177  000000 000000 000567 001   0.
 59   2   2       1730B  READ     +DB   200 020737  000000 000003 112172 001 407.
 34   1   1       8192W  READ     MAM    03 040023  000000 000000 033174 001  24.
 07   2   2        132W  WRITE    MAM    03 013023  000000 000000 002272 001   2.
 87   1  *DIRC     334W  READ     SEG    24 000311  000000 000000 000637 001   0.
 59   2   2       1730B  READ     +DB   200 020740  000000 000003 112172 001 407.
 34   1   1        507W  READ     SEG   407 010662  000000 000002 152770 001 322.
 59   2   2       1730B  READ     +DB   200 020741  000000 000003 112172 001 407.
 87   1   1        128W  READ     +DB   311 003062  000000 000000 116704 001  70.
 34   2   2        499W  READ     SEG   407 004642  000000 000002 113277 001 294.
 59   2   2       1730B  READ     +DB   200 020742  000000 000003 112172 001 407.
 87   1   1        128W  READ     +DB   311 003345  000000 000000 116704 001  70.
 59   2   2       1730B  READ     +DB   200 020743  000000 000003 112172 001 407.
 34   1   1       4956W  READ     MAM    03 060223  000000 000000 026560 001  20.
 34   2   2        141W  WRITE    SEG   137 000102  000000 000005 163206 001 671.
FIN  LDEV          XLOG   FN      REL      WHERE      QMISC  PARM1  PARM2  STAT
 59   2   2       1730B  READ     +DB   200 020744  000000 000003 112172 001 407.
 34   1   1        373W  READ     SEG   206 001130  000000 000001 025574 001 133.
 59   2   2       1730B  READ     +DB   200 020745  000000 000003 112172 001 407.
 87   1   1        128W  WRITE    +DB   311 003345  000000 000000 116704 001  70.
 59   2   2       1730B  READ     +DB   200 020746  000000 000003 112172 001 407.
 34   1   1        296W  WRITE    MAM    03 012223  000000 000000 035400 001  26.
 87   1   1       2932W  WRITE    MAM    03 144623  000000 000000 034564 001  25.
 34   1   1       1924W  WRITE    MAM    05 000023  000000 000000 034230 001  25.
 87   1   1        544W  READ     MAM    03 012223  000000 000003 057106 001 383.
 87   1   1          0W  FOPEN    +DB   311 000000  000000 000000 000000 001
 87   1   1       9260W  READ     MAM    05 000023  000000 000002 134763 001 310.
 37   2   2        896W  WRITE    +DB   321 176166  000000 000005 162623 001 670.
 87   1  *DIRC     384W  READ     SEG    24 001177  000000 000000 000141 001   0.
 59   2   2         80B  READ     +DB   200 015757  000000 000003 041252 001 370.
 34   1   1        373W  READ     SEG   206 005074  000000 000001 025577 001 133.
 87   1  *DIRC     364W  READ     SEG    24 000311  000000 000000 002666 001   2.
 87   1   1       9136W  READ     MAM    04 027623  000000 000003 020170 001 355.
 87   1  +DIRC     384W  READ     SEG    24 001177  000000 000000 000556 001   0.
 34   2   2        232W  READ     MAM    01 177023  000000 000000 007612 001   6.
 37   2   2        116W  READ     MAM    02 131623  000000 000000 007622 001   6.
FIN  LDEV          XLOG   FN      REL      WHERE      QMISC  PARM1  PARM2  STAT
 87   1  +DIRC     384W  READ     SEG    24 000311  000000 000000 000563 001   0.
 59   1   1        507W  READ     SEG   407 001626  000000 000002 153020 001 322.
```

Figure 2.

Sample IOSTAT4 Output

However, External Pointers are not the only area to attack. The number of Buffers available to Image is also a critical factor in overall performance. Increasing the number of buffers via the BUFFSPECS option in DBUTIL.PUB.SYS is generally a good idea as it increases the likelihood that the record that IMAGE needs is already in memory. This improvement can be phenomenal for very small databases or very complex ones. For small databases, most of the "active" records may fit in the IMAGE buffers and so almost no Reads will occur. For complex databases, the number of buffers needed for a single intrinsic call can be as many as 65. While this theoretical maximum is not likely to occur it is very probable that more than the default number of buffers will improve performance. So why not set the number of buffers to the maximum (99)? Glad you asked! There are at least three reasons why not to...

1. Buffers require that precious commodity called Real Memory. Using up memory for IMAGE DBCB's may cause more swapping and those buffers may wind-up on the disc which is not what we want.

2. The more buffers IMAGE has, the more CPU time that must be spent checking to see if the block we want is already in the buffers. This was measured, for a serial read application (DBLOADNG), to be a variance of six percent more CPU time for maximum buffers over the minimum. Obviously, this could mount up on a busy system where the data we want is very unlikely to be in the buffers.

3. There are dynamic areas in the DBCB that grow bigger under certain conditions (locking or opens). The area in the DBCB that is left after the buffers are allocated may not be enough for your application (it wasn't for several of ours).

The number of buffers you decide on should be SET BUFFSPECS=nn(1/120) so that the same number will be used for all users to avoid DBCB expansions. HP seems to recommend 15 to 25 buffers while others suggest 32 for an average application (using BLOCKMAX=512).

There is a direct trade-off between the number of buffers and the size of buffers. As we increase the BLOCKMAX or buffer size, we reduce the maximum number of buffers that IMAGE can fit into its DBCB. Many other factors also serve to complicate the decision to use a larger BLOCKMAX. Larger blocks tend to be more efficient in disc space utilization and so slightly reduce disc needs. Certainly, fewer disc reads happen in sequential access modes. But the longer average access time can serve to slow down writes.

I believe that for many applications the default BLOCKMAX of 512 is NOT a good choice and generally recommend a size of 1024 or larger for database-bound applications. Frequently, the maximum BLOCKMAX of 2048 seems to be the best choice for the systems with many reports or long chains. There must be an application out there that would benefit from a smaller BLOCKMAX but it hasn't found me yet.

Increasing the BLOCKMAX has a pleasant side-effect of increasing the Blocking Factors chosen by IMAGE. Unfortunately, IMAGE does not always pick what we would like for a blocking factor. Its choice is biased towards minimizing disc space and we want better performance! Some have solved this problem by using a feature of ADAGER (Alfredo Rego's legendary utility) to reblock the datasets. I am here to tell you that HP has a way by which you can select your very own Blocking Factor! Deep in the undocumented options of DBSCHEMA.PUB.SYS is "CAPACITY: ccccc(bf);" where bf is any blocking factor that will produce a block within the BLOCKMAX. HP says that they have no plans to document this feature, but, they also have no plans to remove it. We are using it HEAVILY.

Forcing Blocking is very useful when large BLOCKMAX's are employed and several small datasets get small blocking factors. In many cases, the entire dataset can be made to fit into a single block! In the case of large datasets, the increase in blocking factor by one to two is often worth the slight increase in disc space required for the performance improvements gained.

Occasionally, the placement of datasets on specific disc drives will improve system performance. While this would seem to be an easy thing to do, it is not. Deciding which of twenty or thirty datasets should be on what drives in what combinations is nearly impossible. The easiest approach is to use FILERPT to make sure that the most frequently accessed datasets are not on the same drives. Also, related datasets (Master-Detail pairs) may be separated. Any gains made by this technique must be carefully watched as they are easily undone with time, new programs, and different access patterns.

The choice of Item datatypes (particularly Keys) is another area that may severely affect application performance. It is possible to lie to IMAGE about the true datatype of a key in order to have it use a different hashing algorithm but this generally causes problems with QUERY and other utilities. It is generally best to use the datatype that requires the least space (I2 over P10, P12 over Z12). A secondary consideration may be the computational efficiency of a datatype (Subfields may be extracted from P-types faster than from Binary-types). Unfortunately, this paper is about improving performance without changing your programs and changing datatypes means changing programs.

For a true life example, a TRANSACT/VIEW application was tested with various BLOCKMAX and Blocking Factors. Figure 3 shows the detail data related to the tests. The effect of more records per block is that fewer disc accesses take place until the IMAGE buffer limits are met and a performance plateau is reached.

Figure 3.

Disc Accesses as Blocking Factor Varies
BUFFSPECS = 16(1/120)

BLOCKMAX=	512	1024		2048	
	Blocking Factor				
	Normal	Normal	Forced	Normal	Forced
A	4	7	11	17	22
B	21	37	42	85	85
C	2	4	5	7	11
D	3	5	6	11	13
Total disc Required	23599	22610	24066	21201	22764
Total I/O Required	183	173	151	---	153

One hopes that the preceeding generalities may serve to cast a different light on the possible ways of improving IMAGE application performance. We must not forget that "All generalities are specifically Untrue".

It would also serve to remember that these Global optimizations have a very large leverage as all programs are effected. While improvements to individual programs may easily provide more drastic results, many such separate efforts may need to be put together to achieve the same overall results. Also, the amount of effort expended is generally measured in hours for these Global optimizations rather than days.

For Further IMAGE Optimizations

Matheson, Wendy IMAGE/3000: Designing for Performance & Maintainability; 1983 Montreal IUG Proc.

Guerrero, Jorge Image Introduces an End to the Broken Chain; 1982 Copenhagen IUG Proc.

Greer, David IMAGE/COBOL: Practical Guidelines Journal of HP IUG, Vol. 6, No. 1

HPUG Edinburgh 1983 - John Sanders

A Comparative Study of RAPID and COBOL

1. Introduction.

This paper looks at the performance of RAPID and compares it with the COBOL language. It commences with a short history of Data Processing, and reviews RAPID's use of the Stack and register, and contrasts this with COBOL's Data Division, examining the different methods of space utilization.

The potential effects of RAPID on the HP3000 are then examined in terms of response rate and transaction rate, and the methods of measuring this are outlined. The results which were obtained are shown as graphs, where the differences between the two languages can be seen by direct comparison. An attempt is also made to "define" a typical DP Development department, and using this theoretical model, the expected benefits and disadvantages are discussed.

The opinions expressed in this paper are the personal opinions of the author, and in no way reflect Hewlett Packard's position or views.

This paper is not intended to be a scientific treatise. It is aimed at the DP Manager who has either bought, or is considering the purchase of RAPID, and needs to know if that decision is economically valid financially and also wishes to anticipate the effect on an existing or proposed CPU.

2. History of Information Retrieval.

Computing as we know it started life in the early 1960's. At this time, commercial and technical computing were separated much as they are today, but were concerned primarily with the use and effectivity of programming languages, which were structured according to the needs of the application.

Commercial computing relied upon the expertise of a highly qualified, highly paid programmer, whose raison d'etre was to design and code business data systems for data capture and retrieval. However, techniques for the capture and retrieval of data were primitive, and since utility sub-programs were in their infancy and not always available for the functions required, Systems Development required the expenditure of many man-hours programming, often "re-inventing the wheel". Information retrieval was therefore expensive, slow in development and Systems were not likely to be available before the User's requirements had changed.

By the end of the 1960's, beginning of the 1970's, commercial computing had progressed to the stage where much more emphasis was placed on the development of data storage systems, and we now had the facilities of indexing and structure in addition to the simple sequential access methods. These served to improve the speed at which systems could be developed, and also allowed Data Processing Departments to employ less qualified people since the process of "Programming" had been simplified by the introduction of these new file methods. At the same time, we saw the emergence of a desire to integrate these access methods into a simpler, more useful technique which tended to be called "Database" access.

As the 1970's progressed, so did the ability of the industry to provide computer departments with reliable database methods. HP's contribution was with a method which we call "IMAGE". This was, and still is very popular. This provided the computer departments with increased effectivity and reduced production costs, but did little for the end-user since "Programming" was still required. The ambitious HP End-User had an advantage over others in that "IMAGE" was supplied with an enquiry tool called "QUERY". This allowed some interrogation of databases at the cost of a small amount of learning.

Now in the 1980's, we have "RAPID". This provides that vital "Programmer" and "end-user" interface to reduce the requirement for programming for routine enquiry into data stored on your computer. (See Figure 1 for an indication of Rapid's importance to HP).

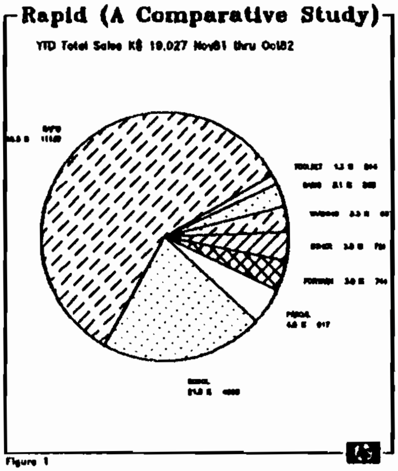


Figure 1

3. RAPID Modules.

RAPID consists of 5 modules. DICTIONARY, TRANSACT, REPORT, INFORM and the RAPID processor. These may be purchased together or individually, with the proviso that the minimum purchase usually includes DICTIONARY/3000. This is necessary because each sub-system uses the information included within the DICTIONARY to gain access to your data. Therefore, a DICTIONARY should exist on each computer which is able to run RAPID. This DICTIONARY will be used unconciously by both End-User and Programmer. N.B. It is possible to run TRANSACT without a Dictionary if all Data Items used are declared in Define Statements. If you are an End-User, then you will want to use INFORM. This provides you with the ability to interrogate data stored on the system without the need for programming, or indeed any computer knowledge other than that required to log on to the system. If you are a Programmer, then you will want to use TRANSACT and REPORT. This provides you with the ability to write programs at a higher level than ever before at an efficiency which reduces development time by up to 30% by eliminating the need to re-specify access methods for your files. Also as a Programmer, you will be pleased to know that all of the facilities that were available to you before are still there, and interface to RAPID very nicely. Such things as use of SL's, Process Handling and special capabilities are all available within RAPID. If you are a not concerned with application development and run a production environment, you will want the RAPID Processor. This will allow you to run the applications developed on other machines. Figure 2 shows the processing sequence between the source code and the final running program, which applies to systems with compilers only.

Thus, RAPID addresses the COMPUTER USER independently of his expertise in the computer field, i.e. the Professional Manager, Accountant, etc. can concentrate on his real business all of the time.

The following notes reflect the Marketing material for Rapid, and serve to introduce the individual modules:

3a. DICTIONARY

DICTIONARY/3000 is a Hewlett Packard facility that provides a comprehensive set of software programs that allow you to:-

1. Create any number of data dictionaries.
2. Create and maintain entries in the dictionary.
3. Create and maintain an IMAGE database.

These programs provide for all phases of System Development, Production and Maintenance.

DICTIONARY/3000 provides an easy-to-use, interactive program for the creation and maintenance of entries in a data dictionary. The program provides a set of commands that you use to enter definitions and to maintain the entries in a data dictionary.

The data dictionary is more than just a dictionary. It is also a directory telling you where data is stored, who is responsible for the data, and much more.

The entries in DICTIONARY/3000 data dictionary can define and describe an organisation's structure, identify the data used by an organisation, tell where the data is stored, identify what programs generate the data, and define and describe INFORM/3000 menu groups to report the data.

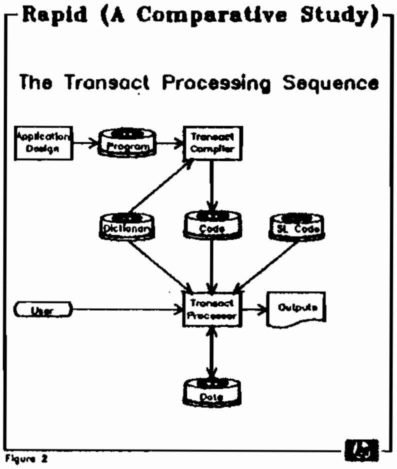
Thus, DICTIONARY/3000's data dictionary is a central repository for information about an organisation and the data processing environment.

DICTIONARY/3000 also provides a complete set of utility programs that allow you to create and maintain an IMAGE database, and to transfer the definition of an existing IMAGE database into the DICTIONARY/3000 data dictionary.

DICTIONARY/3000 utilities support all the functions of a database administrator, such as, loading and unloading of IMAGE databases, and auditing of IMAGE databases.

3b. TRANSACT

Transact Programming Language (TPL) is a high-level computer language. TPL is compiled by the TRANSACT/3000 compiler whose output is then executed by the TRANSACT/3000 Transaction Processor. Both work in conjunction with DICTIONARY/3000 to implement information processing systems unparalleled in efficiency and effectiveness. TPL is used in a wide variety of applications, including manufacturing, finance, and service, and it is used in a variety of industries, including electronics, communications, banking, oil, forestry, and entertainment. TPL users range from computer scientists, who design and implement sophisticated information management systems, to non-technical managers, who want answers to specific questions immediately.


Rapid (A Comparative Study)

The Transact Processing Sequence

Figure 2

3c. INFORM

INFORM/3000 is the End User's tool which allows him to interrogate data without knowing it's structure and organisation. Some grouping of information may be required to be defined within the dictionary, but generally, the user needs to do no more than a "Fill in the Form" type process to obtain information from the computer system, or systems, to relate to his particular vantage point. The important thing about INFORM/3000 is that no computer knowledge is required to allow the user to gain access to his information.

3d. REPORT

REPORT/3000 is a product that allows you to generate reports, both simple and elaborate. In addition to data listings, the report may contain, among other things, summary information, headings and calculations. Data definitions specified in a REPORT/3000 program come from two sources:- DICTIONARY/3000 and/or user-supplied specifications using REPORT/3000 DEFINE and ACCESS statements. REPORT/3000 is a high-level report writing language, report compiler and report producer, and uses Transact/3000 instruction syntax. REPORT/3000 can be correctly thought of as a programmers tool, providing the ability to produce reports of a precise nature, e.g. specifying the exact contents of a given location on the page, and thus provides very precise formatting capability.

4. Transact in the Development Environment.

There are many views as to the use of RAPID in program development. They range from the committed enthusiast, to the outright disbeliever, who propose either that RAPID may be used for all systems, even the largest ones, or that the product is not fit for the development world at all. Obviously, the truth is somewhere in between these two extremes. One thing that can be said of RAPID is that the expections of the majority of users and potential users are incorrectly set, and it is this by and large which causes the wildly varying views.

RAPID was produced to allow simplistic programmatic control of other HP products such as Image, KSAM and V/Plus, using the intrinsic interface in the same way as any other program language would control components of the Fundamental Operating System (FOS). This makes it great for data file creation, modification and deletion. RAPID was not produced to calculate numerical results requiring high precision, nor was it produced to calculate resultant values from arrays of figures. This, therefore, precludes engineering or actuarial applications from the RAPID domain, but makes it ideal for the majority of file applications such as Inventory, Financial Accounting, Order Processing, Personnel and Administrative record keeping, in short, most commercial applications.

Traditionalist development organisation is concerned with the control of the system and its component programs. Data is conceptually a structure providing an interface between systems and programs, and in itself causes the perpetuation of many operational problems, usually because of the difficulty, costliness or timeliness of change. In RAPID, the emphasis is very much on the fundamental data item, or more correctly, data element, and recognises it as a resource to be managed. This encourages the designer to start "at the bottom", (do not confuse with standard structured design techniques), using his data item as a building block, from which he "assembles" his application.

If one starts systems development from the "bottom up", instead of "top down", an opportunity is provided to the designer to qualify the necessary attributes to the data element to provide, at the elemental level, complete qualification as required by the environment, the sizing and typing for its operational use, the classification for organisation and management, and the documentation providing further information, but including a where used list. This has many benefits and advantages to the designer in that duplication of data is avoided, and file generation can be a process of selection of Data Elements from a Company Dictionary, and this often leads to the elimination of redundancy in the elements of a file record. However, this only helps where standards require some assistance and is no use in well managed systems. However, well managed systems are usually young systems, and old systems usually need all the help they can get.

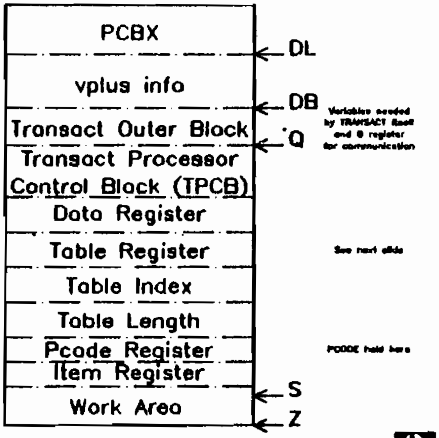
Because RAPID starts at the bottom, the definition of any given data element tends to act as a standard. In the course of time, that element percolates through the organisation enforcing that standard into each system or program of which it becomes a part.

5. How Does Transact Work.

Transact is an interpretive language. Instead of converting from the primary source code at run time, it requires an intermediate compile step which converts the source into a form which is syntactically correct and has some external references resolved. It is therefore analogous to Basic/3000 when executing from a "FAST" file, but here the similarity ends. When data is processed by a Transact "program", the "program" which the user believes is executing is in fact a data segment of another program, "Transact" itself. This is not so different to the previously normal method of executing instructions within a CPU. The difference is, that whereas the CPU is using the code segment as its source of instruction in say, a Cobol program, RAPID replaces the CPU with the Transact processor, and the code segment by a data segment containing both code and stack. So what's new? Its like interpretive Basic/3000. Yes? No!! Basic/3000 interprets the code and emulates the machine language which a compiled program would normally produce. In performing this way, no consideration is given to the needs of the application. RAPID on the other hand, is designed to perform application needs, not instruction set functions. RAPID is therefore capable of performing the execution of many machine instructions through the interpretation of one master transaction instruction.

Rapid (A Comparative Study)

Transact Program Data Stack Layout



PCBX

DL

vplus info

DB

Transact Outer Block

Q

Variables needed by TRANSACT itself and Q register for communication

Transact Processor Control Block (TPCB)

Data Register

Table Register

See next slide

Table Index

Table Length

Pcode Register

PCODE held here

Item Register

S

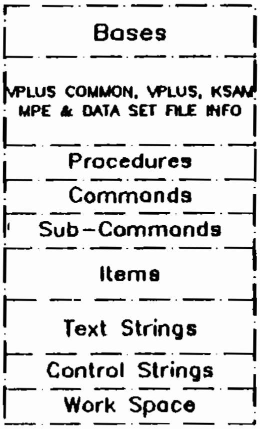
Work Area

Z

Figure 3

Rapid (A Comparative Study)

Table Register



Bases

VPLUS COMMON, VPLUS, KSAM
MPE & DATA SET FILE INFO

Procedures

Commands

Sub-Commands

Items

Text Strings

Control Strings

Work Space

Figure 4

This gives a sequence of instructions which do not just expand one instruction into many, but also include loops for mandatory or optional repeat functions, error warnings and aborts, links to a standard operational environment and many conditional exits which allows the programmer to concentrate on the application at the functional system level, and to avoid the repetitive routine programming that seems to be unavoidable in most other programming languages. Figures 3 and 4 show the layout of the stack as used by Rapid, parts of which will be explained under Rapid and Cobol differences (Section 7).

6. Compare COBOL's Data Division with RAPID's List Register.

Good programming for the Cobol program starts with the construction of the Data Division. Here we can organise our data in convenient groups of records and sub-records which we can clarify by level, starting with the smaller level numbers grouping the larger structures, and the larger level numbers controlling the component groups and items within the larger groups. What we have here is an organisation of a memory into discrete areas which are used for specific data items whose individual use does not vary as a rule. In these areas, the fundamental requirements of the commercial program are performed, that is, the input of data from a file, the massaging of that data, and consequential output of the changed data to another file. This is suited to "Top Down" design as opposed to Rapid's "Bottom Up" design by encouraging the designer to think Files first, then Records and lastly Fields. RAPID has no equivalent to the Data Division. It does have a memory space which is controlled by the List Register, and Data Register among other registers, but the List Register uses the principles of the Stack to enable, theoretically, the same area of memory which was used by one data element, to be used by another when the requirement to maintain the first element is finished. This allows RAPID to minimize the Working Stack, between Q and S, and with the in-built use of the ZSIZE intrinsic, the size of the stack need never be larger than is required provided that the design is RAPID rather than COBOL oriented.

How COBOL uses the memory.

Now the fundamental memory data unit of a process is the stack. (Simple Data Segments and Extra Data Segments are gained as required, and can be considered as common to all types of process for the purposes of this paper). The stack can be thought of as an area of memory which is managed by three control processes. The first is the system control over which you as a user have little control. This affects the PCBX and from time to time some parts of that area between Q initial and the current top of stack S. The second is controlled by the language. This area is the DL-DB area into which the V-Plus Globals and com-area extension is placed together with the V-Buffer. Again, you the user have little control here. The third area is controlled by the user, and includes the Global Area DB to Q initial, where the file records are stored, and the Q initial to Z (Top of Stack), which is the working stack where run time values and inter-program parameters are stored. See Figure 5. Now the important point about the run time stack from our point of view is that the areas defined by the Data Division are located in this Global Area between DB and Q, and that the Local Area between Q and Z contain only the current computation values and parameters. Therefore each defined elementary field will occupy space on the stack, whether it is used or not, and the tendency to expand or contract Q to S for Cobol is small, but is dependent upon the application.

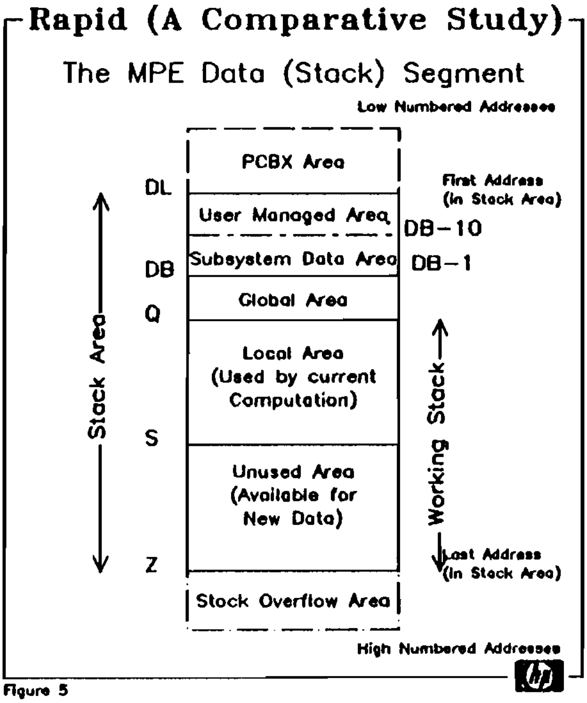

Rapid (A Comparative Study)
The MPE Data (Stack) Segment
Low Numbered Addresses
PCBX Area
DL
User Managed Area
First Address
(In Stack Area)
DB-10
DB
Subsystem Data Area
DB-1
Global Area
Q
Local Area
(Used by current
Computation)
Stack Area
Working Stack
S
Unused Area
(Available for
New Data)
Z
Last Address
(In Stack Area)
Stack Overflow Area
High Numbered Addresses

Figure 5

RAPID memory utilization.

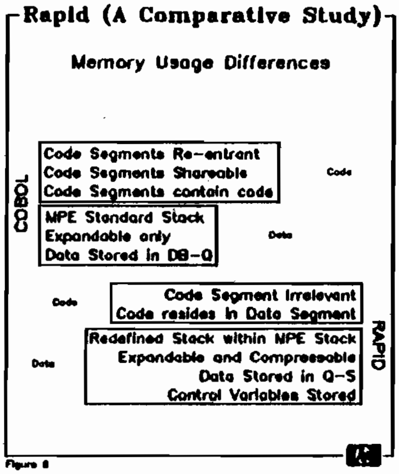
The RAPID stack is configured similarly to the MPE stack, and can often be confusing because of this similarity. The places which are different are those concerned with the control of the PCODE. This includes Pcode control, the Pcode itself and the run time control which also is maintained within the Process Data Segment. Thus, there is an overhead with all RAPID programs within the stack which requires a minimum of 5K words where these associated control variables are stored. This generates a need for more memory in a RAPID application than for the equivalent COBOL program. Then there are differences in the efficiency of machine instruction generation between the two languages, but this is not worthy of consideration in terms of memory requirement, since some RAPID code requires less memory than the equivalent COBOL code, but some COBOL code requires less than the equivalent RAPID. It depends upon what you are doing.

The important thing is, because RAPID tends to require more, typically greater than 20K words compared to about 10K words for an "average" COBOL program, the Data Stack is much more likely to be a candidate for Swap Out, and the Data stack is also more difficult to load. RAPID, however, has the ability to expand and contract it's stack automatically, and can require less room. Unfortunately, this benefit is marred by the MPE mechanism for changing stack sizes which uses a swap to virtual memory to achieve the change. Therefore the reduced memory requirement enforces more memory management I/O which is, on balance, much more expensive in real terms to the User because of the time involved and the conflict with other processes.

Once the actual memory requirement is established, the use of that memory by RAPID is probably superior to that of COBOL. I do not include here any requirement for "number crunching" which RAPID does very badly, about 20 times slower than COBOL, only the concept behind the usage which does not enforce typing or code conversion as a mandatory requirement. Fundamentally, RAPID uses the memory defined as a stack of Bytes. They are organised in Words in practice, since MPE works this way, but it is very easy with RAPID to split and re-combine words in non-logical ways, usually inadvertently, and this can cause problems when testing a finished program if Data Management is not included as part of the "stack" design.

7. Differences between RAPID and COBOL Memory Utilization

Once a COBOL program has been written, the Data Division contains the definition of how the memory will be used, and implicitly, how the run-time stack will behave. RAPID has no similar mechanism which allows for this pre-definition of a memory area except for that area required for Transact's own global variables, and the mandatory pointers. This on average requires 5K words and is an overhead which RAPID enforces, and is one of the first areas created during start up. Then, with a recognisable stack structure set up, the IP code file will be read into the current top of stack 'S', and this will contain information on the Procedures, Commands, internally defined items and the interpreted code held in the PCODE register.



Rapid (A Comparative Study)

Memory Usage Differences

COBOL
Code Segments Re-entrant
Code Segments Shareable
Code Segments contain code
Code
MPE Standard Stack
Expandable only
Data Stored in DB-Q
Data
Code
Code Segment Irrelevant
Code resides in Data Segment
RAPID
Data
Redefined Stack within MPE Stack
Expandable and Compressable
Data Stored in Q-S
Control Variables Stored

Figure 6

The COBOL program file whilst equivalent to the "IP" file, has the ability to store its code in a code segment, thus reducing the space required in the data stack, but not the total requirement for memory. This also has the added advantage of allowing the code to be re-entrant, i.e. numbers of users can execute the code from different points, independently, and provides for savings of memory as the numbers of users increase, since only one copy of the code is ever required. RAPID must always hold a unique copy of the code for each user, thus there is an inefficiency of memory usage for increasing numbers of RAPID users. During operation, there are differences again. COBOL uses data items defined in the DB-Q area which reside in fixed locations. The same items in RAPID, are laid down as required, and will appear in the Data Register. This will be found in the Q-S area in the stack, see figure 3. Thus whatever space is defined by COBOL during programming, that space is required at the stack for the duration of execution. Items defined in RAPID do not require space until they are required for use and the space requirement can be removed when it is no longer required. Finally, COBOL does not automatically adjust its stack at runtime. Once defined and prepared with given stack or maxdata values, those are the values which will generally apply throughout the execution of that program, unless specific stack size measurement and adjustment is included by the programmer. Expansion of the stack is automatic up to maxdata through normal MPE operation, but reduction is not. RAPID has calls to ZSIZE built in, therefore, RAPID can not only take advantage of MPE, but also can to a limited extent, adjust its stack to suit the space required at run time. Figure 6 shows the main differences between the two languages.

8. Affects on the processing ability of the HP3000.

Comparative Performance by Response

From a paper by Roy Martin, September 13th, 1978 we can estimate the transaction response time of the processor from the simple equation.

$R = Q \times C$

where Q is the average queue length
and C is the average transaction completion time

Q is calculated by $T - (M/C)$

where T is the number of terminals
and M is the think time per transaction

In a specific measured case, think time was measured at 10 seconds, transaction completion time was 0.5 to 0.7 seconds and response time was estimated from the above equation to be nil until more than 14 terminals were active. The actual response time when measured on a small machine was found to start increasing from 4 terminals, but this was shown to be due to the process under measurement not being memory resident. When they were memory resident, the theoretical model provided a good fit with the actual results. Using the results developed in Roy's paper provides us with a base line which looks at specific bench marks. We can therefore take the "think time"/"transaction completion time" ratio as a universal measure of performance in terms of response, regardless of what language the transaction is written in, and reduce the problem to a simple measurement of the time required to complete a given process.

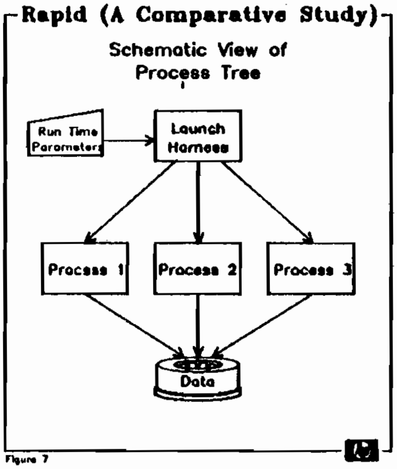
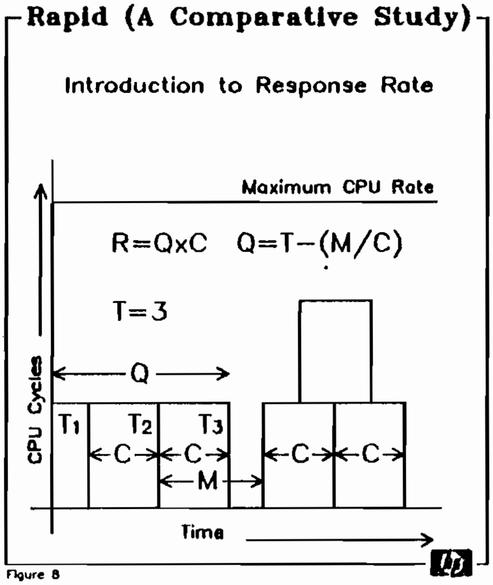


Figure 7

The measurement of the time required to complete a given process is fairly simple to achieve. See Figures 7,8 and 9. Simply a question of preparing a RAPID program and a COBOL program to perform the stated process, and then running each on a dedicated machine, i.e. on their own, using the same configuration in each case, then measuring the time taken, preferably over a finite number of repeated processes. To be scientific, one must perform this operation for each discrete type of process possible, and evaluate the implication of loading, unloading, virtual memory functions etc. on the process in question, to ensure that the results can be extrapolated for any general multi function process. For a commercial evaluation such as this, this rule does not need to be so rigourously applied, since an answer in absolute terms is not required. With the transation process identical we can simply measure the time taken to complete one cycle (simulating one terminal), then run the process in conjunction with itself (simulating two terminals) and so on till we have enough processes running to limit the response time and build up a wait queue. The thinking time within the process is achieved by simply performing a non-effective loop until the think time has elapsed. The transaction itself was structured to use image calls to allow the addition of Detail records to a database using automatic masters for simplicity, and measurements taken as above for both COBOL and RAPID. This gives a series of values indicative of a transaction using DISC I/O. Then removing the Put statements, a measure of the processing exclusive to the disc I/O can be obtained.



Rapid (A Comparative Study)

Introduction to Response Rate

Figure 8

Rapid (A Comparative Study)

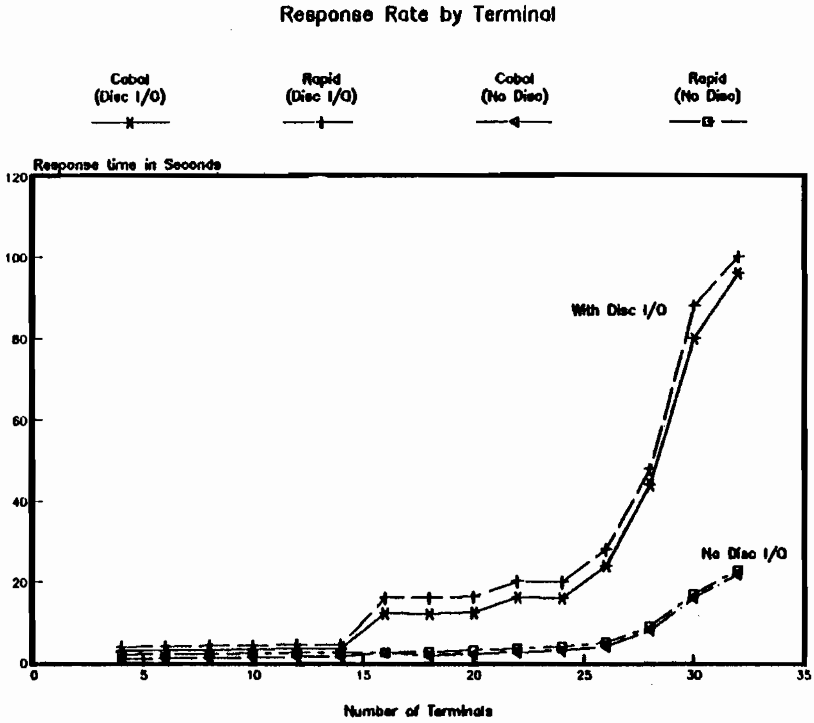
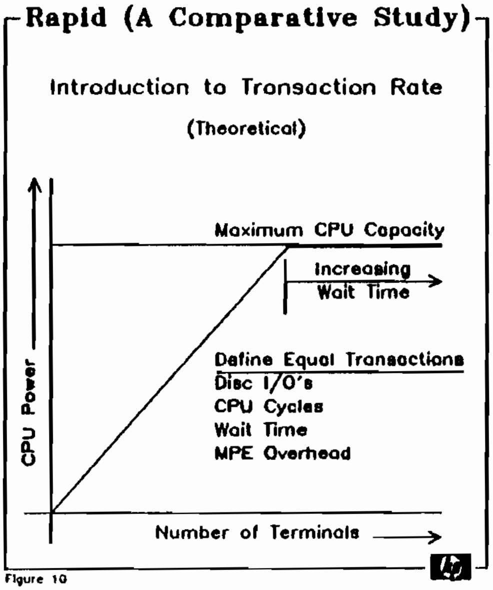


Figure 9

Comparative Performance by Transaction Rate

All performance measurement is tricky and is dependent upon the specific conditions under observation at the time. The measurement of transaction rate is even more difficult since one must arbitrarily decide what constitutes a transaction. Further, one must qualify the transaction as light, medium or heavy. The theoretical transaction rates for a perfect machine are shown in Figure 10. The actual results and summary of losses when compared to Cobol are shown in Figures 11 and 12 respectively.



Rapid (A Comparative Study)

Introduction to Transaction Rate

(Theoretical)

Figure 10

For the purpose of this study, I used a direct read from an MPE editor file and an Image add to a detail data set with automatic masters as before. However, the disc I/O is aimed at producing 5 I/O's per second per transaction, about what is required to add 1 record to the data set. Each process performing the transaction will prefix the key field with an identifier based on the PIN number and will serve to make each detail record added unique by process. Then, using a similar launch harness as was used for response measurements, we can attempt to saturate the CPU and measure equivalent transactions per hour for both RAPID and COBOL. At the point where the CPU begins to saturate, we can obtain a measure by number of processes (or terminals) of the transactions per hour for the given transaction that the specific CPU can tolerate, and we can determine the approximate differences for a RAPID or COBOL program. If we try to load the CPU beyond this point, we can expect the effective maximum rate to fall, due to the increasing overhead of MPE.

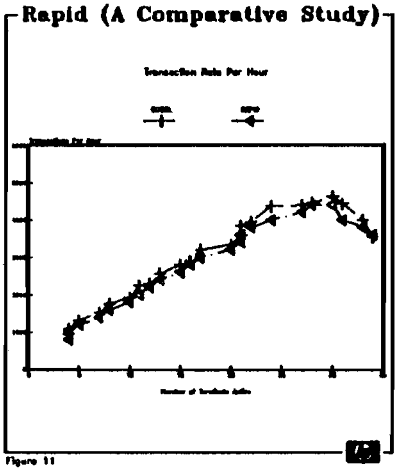
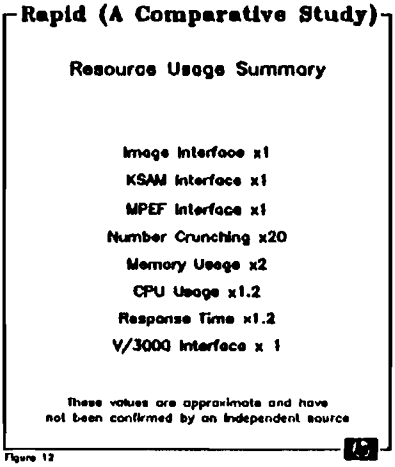


Figure 11



Rapid (A Comparative Study)

Resource Usage Summary

Image Interface x1
KSAM Interface x1
MPEF Interface x1
Number Crunching x20
Memory Usage x2
CPU Usage x1.2
Response Time x1.2
V/3000 Interface x 1

These values are approximate and have not been confirmed by an independent source

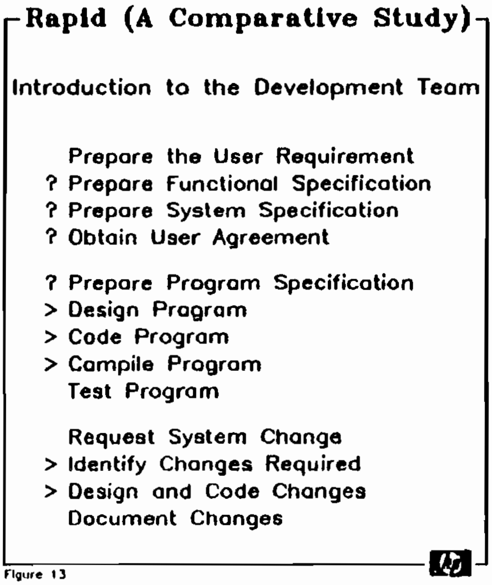
Figure 12

Performance of the Development Team

I can not set up a simple benchmark to indicate the effectivity of one language over another. Apart from the implicit design considerations which apply to all languages (there is no substitute for thinking), there is the added impetus gained from familiarity and from preference. If I were personally left to program in any language, I would choose PASCAL, but if I had to develop within a budget, I must choose RAPID. The differences are governed by esoteric satisfaction in the language itself, or by the demands of the economic world in real or wooden dollars. To take a simple example, the production of a report, led us to look at the steps involved in writing this in COBOL. These would be as follows:-

1) Design the layout	1 day	
2) Agreement from the user		2 hours
3) Prepare program specification	1/2 day	
4) Program/Code/Input/Compile	2 days	
5) Test	1/2 day	
6) Approval from the user		2 hours
7) Implement		2 hours
	4 days	6 hours

Figure 13 shows a typical development sequence indicating the areas where Rapid can improve the performance of development.



Rapid (A Comparative Study)

Introduction to the Development Team

Prepare the User Requirement
? Prepare Functional Specification
? Prepare System Specification
? Obtain User Agreement

? Prepare Program Specification
> Design Program
> Code Program
> Compile Program
Test Program

Request System Change
> Identify Changes Required
> Design and Code Changes
Document Changes

Figure 13

Now on the surface, RAPID offers very little advance over the above except in functions 3 and 4. Even a 100% saving here offers only 50% saving overall. Obviously this maximum saving is not physically possible, but, for the average but simple report, RAPID requires 1 statement to "OUTPUT" the report, and one statement to satisfy format requirements. Compare this to the 200 line program required by COBOL, and you can see that large savings in the coding functions of 3 and 4 alone are indeed possible. But, since the function of the systems analyst is to provide the interface between the user and the computer staff, and since the act of programming has been so greatly simplified, it now becomes possible to allow the analyst to produce reports directly from the initial discussions with the user and to bring the user much closer to the sharp end of programming so it is thus possible to reduce function 1 by 50% and function 5 almost entirely, assuming of course, that sufficient valid data is available from the start. Therefore, assuming a 50% reduction in those functions the steps involved in writing the report in RAPID are as follows:-

1) Design the layout with the user		3 hours	*
2) Agreement from the user		2 hours	
3) Prepare program specification		1 hour	*
4) Program/Code/Input/Compile	1 day		*
5) Test		2 hours	*
6) Approval from the user		2 hours	
7) Implement		2 hours	
	2 days	4 hours	

This compares favourably with COBOL's 4 days 6 hours.

Now it is true that the reporting abilities of RAPID are superior by far to its other functions, and results like those above are very easily obtained in most cases. Further, the abilities of INFORM, which places the power of the CPU directly into the users hands, provide a very powerful means of "offloading" much of the DP departmental dross of utility reporting to the user departments where I believe it should be. However, with such a tool, it is possible for the user to request a 50,000 line extract, sorted and printed, which may be at best unavoidable, but is at worst wasteful since if the user is uneducated in computer matters, he may have requested the wrong information, and will immediately re-start the process again without using any of the previous output. This is not important provided that the CPU is dedicated to user functions, but often it is a shared resource and there is not too much spare capacity. Two points thus arise. Firstly, because the simplicity of RAPID allows more uncontrollable wastage, it is important that User Computer Education takes a higher priority than before, and secondly, the primary usage of resources using RAPID being the CPU and peripherals, i.e. not labour intensive, the computer budgets should include a provision for larger machines to cover both for the increased usage and also the increased waste. This total increase should be between 10 and 20% of the whole. Note that the labour content although not mentioned, is expected to reduce by 30%. Now assuming an average DP department of 20 people at say $30,000 per head.

(This figure is probably low) we can expect labour savings on the current achievement rate of:-

$$\text{Saving} = (30{,}000 \times 20) \times \frac{30}{100}$$

$$= \$180{,}000$$

or in other words an effective increase in the current department of 6 heads to catch up with backlogs. Figure 14 gives a reasonable "guestimate" of the investment/time differences between the two languages.

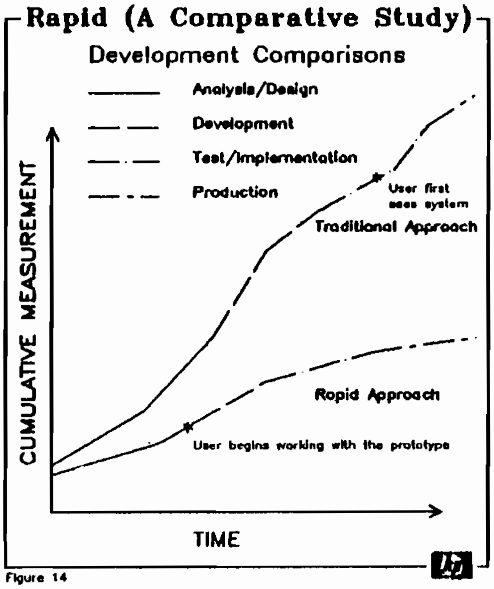


Figure 14

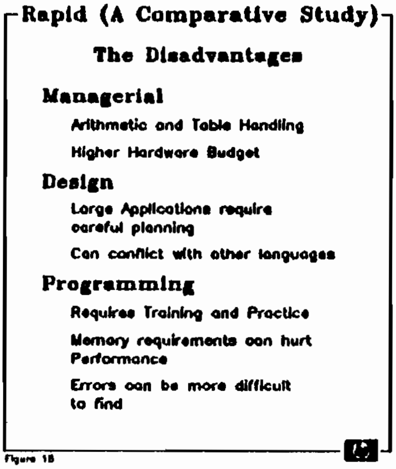
On the negative side, the current expenditure per annum on hardware for the average installation (using current leasing figures) is approximately $200,000. An increase of 20% requires an additional $40,000 to be spent on hardware if the RAPID approach is taken. Compare this with the gain from above of $180,000 and you can see that a 450% return per annum is worth having.

Incidentally, when I refer to the "average installation" I refer to my own experience of development teams in the U.K., who generally use Model 44 machines with 2MB memories.

9. Benefits and Problems.

Problems: Limited arithmetic and table handling.
No use possible of the segmenter. (No Link Edit facility).
Large stack requirement caused by:-
  a) Transact Outer Block Variables.
  b) Holds source code with data.
Changing stack sizes requires Virtual Memory Swapping.
Use of "Load Proc" intrinsics dramatically slows program load times.
Data Bases are locked at Base Level only.
Large applications require planning.
Can be incompatible with existing development in other languages.
Requires a new approach to high level programming which is unfamiliar to most programmers.

Benefits: Saves 30% of development time.
Saves much in development cost.
Good performance if not "number crunching".
Compact.
Automatic Global environment defined.
Simple and powerful De-bugging aids.
Variable Data stacks.
A better approach to Information Management.
Reduced Application Development Costs.
Increased user involvement and satisfaction.



Rapid (A Comparative Study)

The Disadvantages

Managerial

- Arithmetic and Table Handling
- Higher Hardware Budget

Design

- Large Applications require careful planning
- Can conflict with other languages

Programming

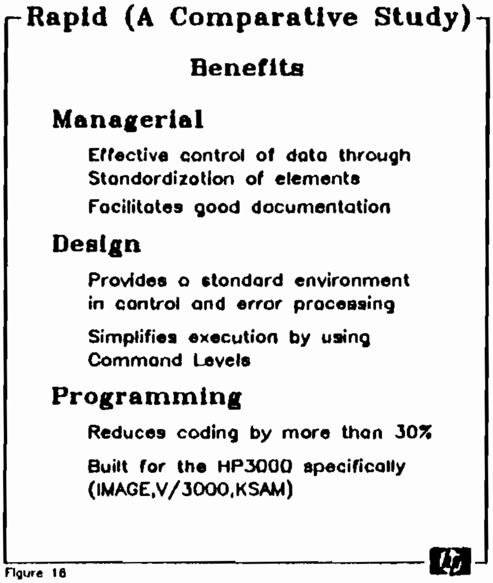
- Requires Training and Practice
- Memory requirements can hurt Performance
- Errors can be more difficult to find

Figure 15

10. Summary

To sum up, it seems that RAPID is not as efficient as COBOL, but then neither is COBOL as efficient as Assembler or Machine Language. It requires more memory and more CPU, and the elapsed time can be longer if compared with a COBOL/RAPID benchmark on the same machine, mainly due to the additional work which RAPID must do in the interpretive phase. On the plus side, allowing for adequate training in RAPID which would require the same time per programmer as for COBOL, (About 6 months), System Development is much quicker, particularly if using Prototype techniques, and this additionally provides benefits in drawing the User much closer into the design and development phase providing faster System Accuracy and Acceptance. Standards are easier to accept and maintain since RAPID builds on Standard Operational Procedures and Error Processing, and during the development and maintenance phase, speed of compilation allows RAPID modification and enhancement.

For the efficiency, Figures 9 and 11 show that the difference between the two languages are so small, (10-20%), that they could almost be ignored, however, in the real world, performance tends always to worst case, and the wise DP Manager would be wise to order 10-20% more hardware. Compare this add-on cost to the savings of 30% in the development budget, (or ability to do more development for the same cost), and one can see that the savings far outweigh the cost.



Rapid (A Comparative Study)

Benefits

Managerial

Effective control of data through Standardization of elements

Facilitates good documentation

Design

Provides a standard environment in control and error processing

Simplifies execution by using Command Levels

Programming

Reduces coding by more than 30%

Built for the HP3000 specifically (IMAGE,V/3000,KSAM)

Figure 16

we are watching the beginnings of a fundamental change I believe in the way in which Development of Software is achieved. Because software is so expensive, it is imperative that reductions in cost in the development phase are achieved wherever possible, and that the power of the computer is used to this end. The primary objective is cost, the secondary objective is simplicity so that the computer is brought closer to the user. Unfortunately, I believe that this signals the demise of the routine programmer. Obviously good programmers will always be required, (for a while), but the current shortage of routine programmers will be met by "Smart" software. RAPID is the first step on this road and for the HP3000, the DP Manager really does not have a choice in whether to buy RAPID or not. It should be present on each HP3000. To quote the marketing literature again, "It'll cost you thousands of dollars to buy this software from us. Or millions if you don't", states what is almost a universal truth. This satisfies the primary objective of cost by successfully implementing the second objective of simplicity. There are one or two problems with RAPID it is true, it is a new product, but those who remember COBOL when it first hit the market place will tell of the far greater problems that COBOL had at the time.

11. References

1. Response Rate by Roy Martin - 1978
2. General Product Marketing Information
3. Product Reference Manuals

IAS/3000

Integrated Accounting System for the HP3000

1. MAJOR FEATURES AND DESIGN AIMS

IAS/3000 is CODA's Integrated Accounting System for the HP3000. It addresses the main functional areas of Sales, Purchase and Nominal Ledgers.

IAS/3000 is designed to operate in an accounts department and to enable the accountant to keep accurate books, control cash flow and to produce management accounts.

Designed for the HP3000

IAS/3000 is specifically designed for the interative database environment of the Hewlett-Packard HP3000 range of business computers.

Designed for Accountants

IAS/3000 is an accounting system designed, developed and supported by accountants for accountants. It is the product of many years experience of managing accounts departments on the part of its principal designer.

IAS/3000 does not require the user to have any detailed knowledge of the HP3000 computer or the internals of the package.

Easy to use

Throughout the development of IAS/3000 careful attention has been paid to ease of use. The system offers clerks and supervisors simple, well controlled and reliable input procedures. It also offers supervisors and managers easy access to information in detail and summary form.

Easy to install

Installation by a CODA accountant is 'bundled' with the package. 4 to 5 days is all that is required to discuss, agree and set up the parameters which will tailor the database for the chart of accounts, documents, ledgers etc you require.

Transferring customer/supplier names and addresses and nominal accounts from your old system is made simple with the IAS/3000 Aids programs supplied with the package.

Interactive

All updating is performed interactively on a document by document basis. No input is accepted unless the document balances. The books are always up to date and balanced.

IAS/3000 is designed to put terminals on the desks of clerks who originate the data. Conceptually each clerk has a copy of the up to date books on his/her desk and can enquire on or make entries to them interactively. We like to think that IAS/3000 takes you back 150 years to the days before batch computers got in between the Accountant and his books.

Integrated Sales, Purchase and Nominal Ledgers

IAS/3000 is a totally integrated accounting system covering the major areas of Sales, Purchase and Nominal Ledgers.

The Sales and Purchase Ledgers are identical - but different. They use the same file format but various programs are optimised to serve the sales and purchase staff. It is possible to buy from a customer and sell to a supplier. It is also possible to pay a customer and receive payments from a supplier.

Up to 99 companies, each with 99 Sales and Purchase Ledger 'Control' Accounts, may be configured within each IAS/3000 database. Any number of databases may be set up on the one HP3000 using just the one copy or licence for IAS/3000.

Other Ledgers

In addition to the 99 Sales and Purchase Ledgers IAS/3000 will also support 26 other Ledger Sections within each of the 99 Nominal Ledgers. These Ledgers support detailed analysis of nominal ledger entries. Typical uses are analysis between Contracts, Projects, Jobs, Parts, Fixed Assets, Employees and Expense Codes.

Books On-line indefinitely

The heart of IAS/3000 is a database of nominal detail entries made to the books. IAS/3000 will retain these entries on-line on disk for up to 99 years subject to the availability of the necessary disk space. A special 'Archive' program will copy over into a separate database all documents older than a certain date; on enquiries the user then has the option to enquire on the 'active' data or the 'active' and 'archive' data. There can be 99 generations of archive.

CODA normally recommend that as a minimum, you have sufficient disk storage for the whole of a financial year plus 3 months until the auditors have completed their work. This eliminates most of the hard work at year end for the accountant. Every entry in the books is instantly available at the terminal and searching through month end printouts is completely obviated.

Comments, Remarks, Descriptions

The handwritten memorandum comments traditionally scribbled on ledger cards etc can be recorded in the IAS/3000 books on line on disk. Each document and each supplier or customer account can have a comment attached to it of up to 64,000 lines each of 30 characters.

Remarks can be entered and printed on a particular run of remittance advices or statements.

A 30 character description is allowed on every individual detail entry in the books.

The IAS/3000 philosophy is that it is much cheaper to store this information on disk and have it instantly accessible than it is to store it off line on paper and filed manually in several places.

Security

Every customer, supplier, nominal and expense account ledger, company and document type can have a security status to prevent unauthorised access. The level of access allowed for each operator can be set at individual function level within a program and at program level. For example an operator can enter cash but not mark it off; an operator is allowed into the mark off program but only to perform one of the 12 available types of allocation (each type is a function within the program).

Budgeting & Accounting Data

When documents are entered into the IAS/3000 database detail book-keeping entries are posted to the appropriate nominal ledger accounts and customer or supplier account. In addition the system adds the entry values to the appropriate period balance and current balance for each nominal account.

A set of period balances is kept for each nominal account for up to 99 years for up to 26 budget codes (A-Z). A is reserved for Actuals, typically B is used for Budgets, F for Forecasts, E for Estimates etc. These sets of period balances are available to be used and reported on by the **Financial Reporting System** which is an optional part of the IAS/3000 package.

The period balances for budgets, estimates, forecasts etc may be set up and manipulated very easily. Individual period balances (up to 18 in the year) may be entered or an annual figure may be phased or spread over the individual periods using one of up to 99 tables of spread factors.

Financial Reporting System (FRS)

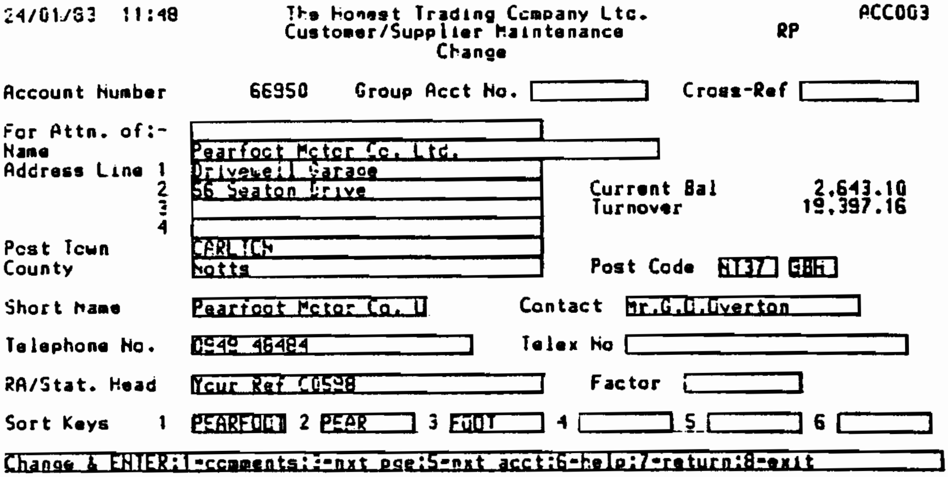
A set of programs enable the Accountant to design, with simple fill in the form screens, his own Final Accounts and Budgeting reports. He can draw on the nominal account data described above and merge nominal accounts together in a number of different formatted reports. He can meet the reporting requirements of head office, company, directors, managers and the accountant.

A set of period end accounts can typically be produced within minutes of making the last real entry in the books or an adjustment to a budget figure. FRS is designed to enable the accountant to produce several sets of accounts within a few hours iteratively approaching the answer he is searching to achieve.

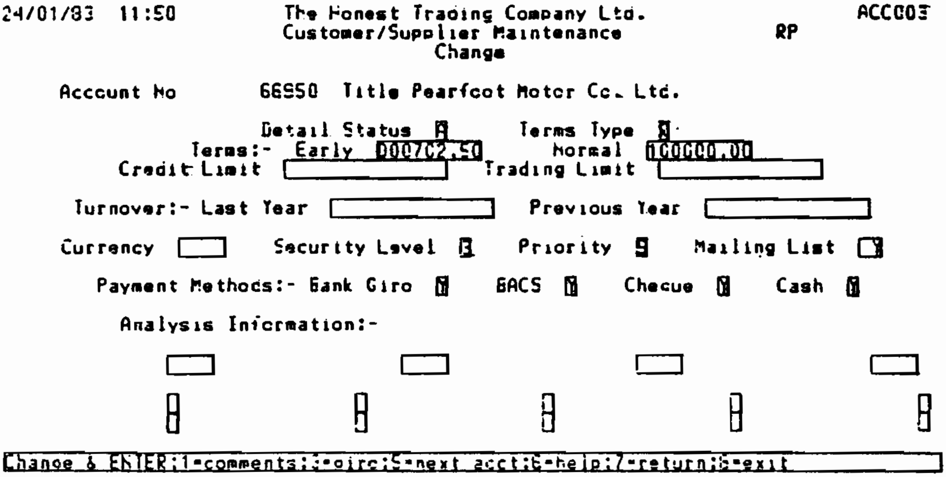
Multi-currency, Multi-lingual

IAS/3000 is installed in a number of European countries. Facilities have been included in the package to conform to the statutory requirements of those countries and foreign language screen and report formats are available.

The base or home currency of the database can be any currency and IAS/3000 has automatic dual currency operation features on import/export accounts and nominal accounts.



```
24/01/83  11:48        The Honest Trading Company Ltd.                ACC003
                       Customer/Supplier Maintenance            RP
                                  Change

Account Number       66950   Group Acct No. [          ]   Cross-Ref [          ]

For Attn. of:-       [                              ]
Name                 [Pearfoot Motor Co. Ltd.                 ]
Address Line 1       [Drivewell Garage              ]
             2       [56 Seaton Drive               ]   Current Bal        2,643.10
             3       [                              ]   Turnover          19,397.16
             4       [                              ]
Post Town            [CARLTON                       ]
County               [Notts                         ]   Post Code [N137] [6BH]

Short Name           [Pearfoot Motor Co. L]   Contact [Mr.G.D.Overton         ]

Telephone No.        [0949 46484          ]   Telex No [                      ]

RA/Stat. Head        [Your Ref C0598                ]   Factor [          ]

Sort Keys        1   [PEARFOOT] 2 [PEAR    ] 3 [FOOT    ] 4 [        ] 5 [        ] 6 [        ]
[Change & ENTER:1=comments:2=nxt pge:5=nxt acct:6=help:7=return:8=exit              ]
```



```
24/01/83  11:50        The Honest Trading Company Ltd.                ACC003
                       Customer/Supplier Maintenance            RP
                                  Change

     Account No        66950  Title Pearfoot Motor Co. Ltd.

                       Detail Status [A]      Terms Type [N]
                 Terms:-  Early [000702.50]      Normal [100000.00]
           Credit Limit [              ] Trading Limit [              ]

     Turnover:- Last Year [              ]   Previous Year [              ]

     Currency [   ]    Security Level [3]    Priority [9]    Mailing List [Y]

       Payment Methods:- Bank Giro [Y]    BACS [Y]    Cheque [Y]    Cash [Y]

           Analysis Information:-

              [   ]              [   ]              [   ]              [   ]

              [ ]             [ ]             [ ]             [ ]             [ ]

[Change & ENTER:1=comments:2=giro:5=next acct:6=help:7=return:8=exit                ]
```

2. PURCHASE LEDGER

The aim of the purchase ledger system is to allow the supervisor and clerks to maintain firm control over the ledger. The effort required to 'run the system' has been minimized.

Many features are included to ensure any supplier discounts available are infact taken and suppliers are paid as late as possible but at the same time avoiding complaints.

The Supplier File

The supplier file contains extensive information about each supplier:-

Account number and optional 'alpha sort keys' which enable the supplier to be listed under and accessed by additional keys as well as the account number. For example the Pearfoot Motor Company Limited may be found by it's account number or 'Pearfoot' or 'Pear' or 'Foot'.

Account numbers for the Group this supplier is a member of, a Cross Reference Account in the Sales Ledger and a Factor Account.

Name and Address information.

Telephone number, telex number, contact name, company short name.

Terms - early and standard payment terms and the rules for their use if a due date is entered for an individual invoice.

Currency marker - refers to a table of exchange rates held in the IAS/3000 database.

Security code - each account has a code to prevent unauthorised reference.

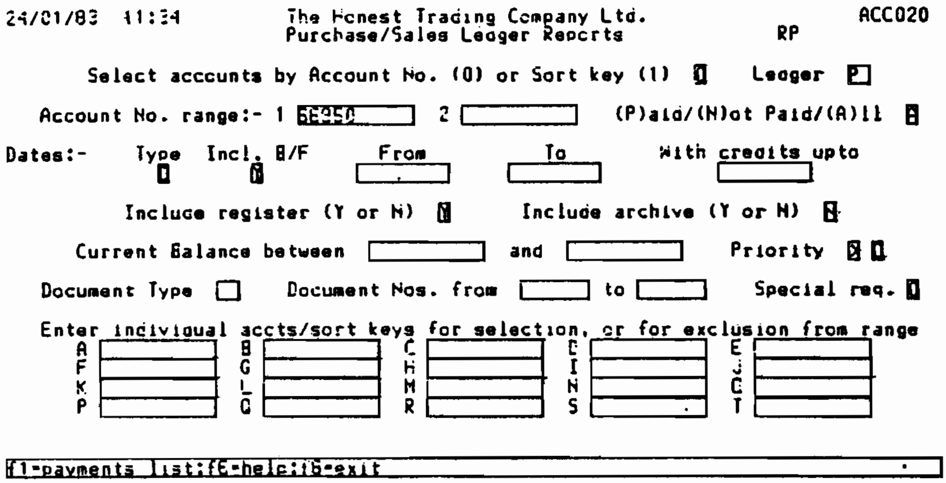
Priority - a method of grouping accounts which may be scattered across more than one ledger.

Method of payment - provision for manual payments and payment by computer of Cheques, Credit Transfers and BACS tape.

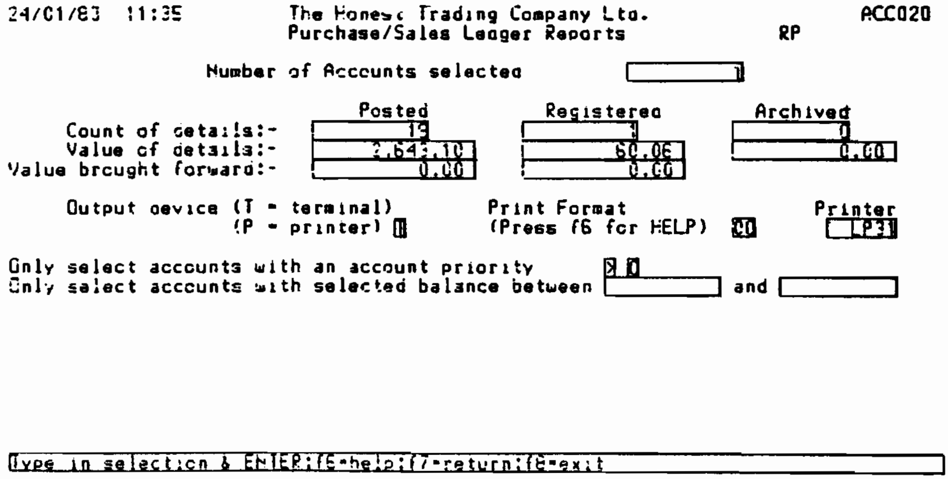
Analysis codes - 4 four character and 10 single character fields can be labelled and used for analysis purposes typically by Order Processing, Sales Analysis suites integrated into IAS/3000.

Credit limit, trading limit, turnover history fields and Current Balance.

Additional user defined fields.



```
24/01/83  11:34          The Honest Trading Company Ltd.                 ACC020
                         Purchase/Sales Ledger Reports              RP

         Select accounts by Account No. (0) or Sort key (1) 0    Ledger P

      Account No. range:- 1 6E950       2           (P)aid/(N)ot Paid/(A)ll A

Dates:-    Type  Incl. B/F    From        To          With credits upto
            0       N

            Include register (Y or N) N     Include archive (Y or N) N

        Current Balance between           and            Priority > 0

      Document Type       Document Nos. from      to        Special req. 0

      Enter individual accts/sort keys for selection, or for exclusion from range
        A               B               C               D               E
        F               G               H               I               J
        K               L               M               N               O
        P               Q               R               S               T

f1=payments list:f6=help:f8=exit
```



```
24/01/83  11:35          The Honest Trading Company Ltd.                 ACC020
                         Purchase/Sales Ledger Reports              RP

                    Number of Accounts selected                 1

                              Posted          Registered        Archived
     Count of details:-            19                 1                0
     Value of details:-      2,643.10             60.06             0.00
Value brought forward:-          0.00              0.00

     Output device (T - terminal)         Print Format                Printer
                   (P - printer) P        (Press f6 for HELP) 00        LP31

Only select accounts with an account priority        > 0
Only select accounts with selected balance between              and

Type in selection & ENTER:f6=help:f7=return:f8=exit
```

Enquiries/Reports

The following reports are available on demand:-

- Remittance Advices
- Ledger Cards
- List of account balances
- Age Analysis
- Proposed Payments Lists

Selection can be on the following:-

- All accounts in all Purchase Ledgers
- All accounts in a Purchase Ledger
- A range of accounts
- A range with up to twenty exclusions
- Up to twenty individual accounts
- A single account

In addition, within the choosen accounts, selection may be by:-

- Input date, Document date, Due date
- Current account balance
- Priority
- Document type
- Status (Paid, Not Paid, All)

All reports, with the exception of the Age Analysis, can be produced either on the VDU or one of Hewlett-Packard's wide range of printers. Examples of Ledger Card, Remittance Advice and Age Analysis reports are shown overleaf.

Purchase Invoice Register

An optional feature of the system supports a register of purchase invoices. This allows invoices to be recorded at time of arrival in a memorandum file. The register subsystem provides reports on registered invoices awaiting approval and entry. It also allows comments to be added to the entries.

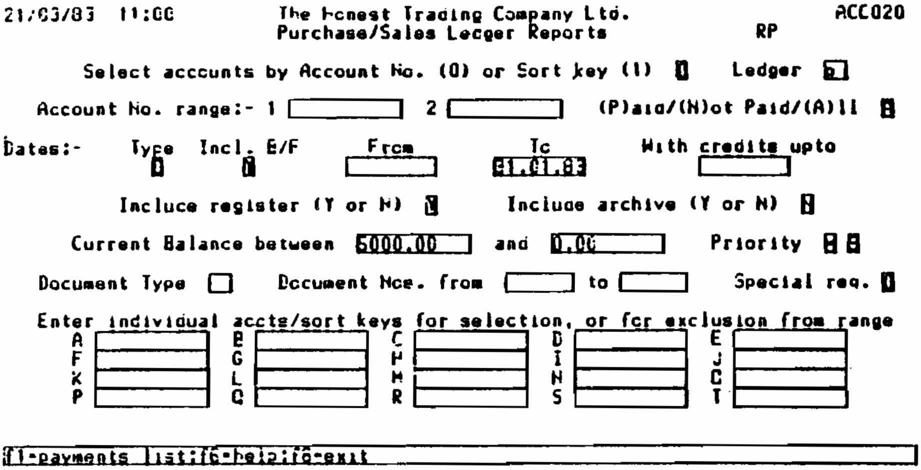
When a registered invoice is entered into the books, it is not necessary to re-input the details on the register. The information registered does, of course, include provision for the supplier's invoice reference and/or a comment.

Any VAT input tax on registered invoices can be reclaimed even though the invoices have not been passed for entry.

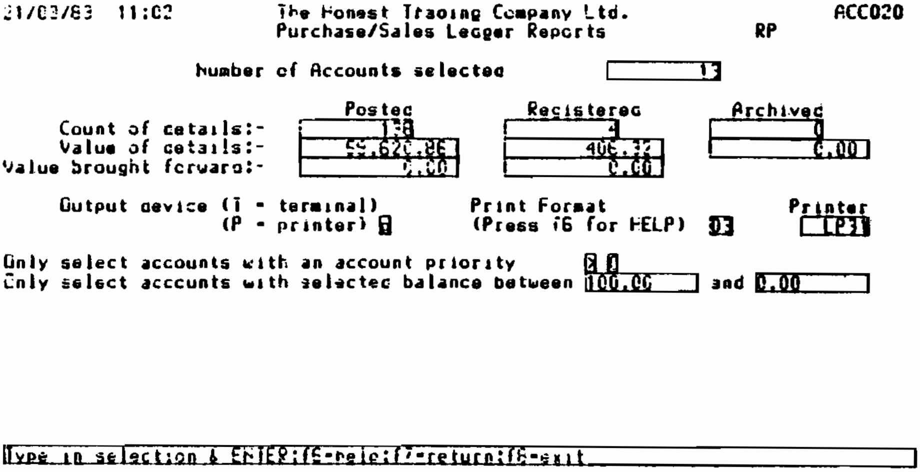
The register invoices can be included on purchase ledger reports thus considerably simplifying the task of reconciling suppliers accounts.

```
10 Mar 02   09:03     The Honest Trading Company Ltd.        Age analysis of Supplier accounts by RP        ACC020    Page No 0001

LN P   B/F Y Akey 0 Reg Y Arc N Pri > 0 St N Sp 0 D2 31.07.88
O/P P Prt    LP31 Pr 02 Pri >0 Ageing M I N 3 D

Account No   Short Name             Total  | Unallocated |  Later  |   Jul 88  |   Jun 88  |  May 88 |  Apr 88 |  Mar 88 | Earlier |   Held  |Comment
                                           |             |         |           |           |         |         |         |         |         |
   66650 Printright                 65.23  |             |         |           |     65.23 |         |         |         |         |         |
                                           |             |         |           |           |         |         |         |         |         |
   66700 Nost. Stat.                47.55  |             |         |           |     47.55 |         |         |         |         |         | Refer all Queries on this
                                           |             |         |           |           |         |         |         |         |         | account to Mr Chapman. Send
                                           |             |         |           |           |         |         |         |         |         | NO Cheques. PHF-18.Jan.88
                                           |             |         |           |           |         |         |         |         |         |
   66750 Vendadrink                 48.30  |             |         |     48.30 |           |         |         |         |         |         |
                                           |             |         |           |           |         |         |         |         |         |
   66850 James Harris           28,926.16  |             |         | 28,926.16 |           |         |         |         |         |         |
                                           |             |         |           |           |         |         |         |         |         |
   66851 BSC Plates              8,823.83  |             |         |  8,823.83 |           |         |         |         |         |         | This account MUST be paid
                                           |             |         |           |           |         |         |         |         |         | each month by the 20th.
                                           |             |         |           |           |         |         |         |         |         |
   66852 GKN South Wales HOME   18,747.99  |             |         |           | 18,747.99 |         |         |         |         |         |
                                           |             |         |           |           |         |         |         |         |         |
   66900 Ringwell                1,465.09  |             |         |  1,465.09 |           |         |         |         |         |         |
                                           |             |         |           |           |         |         |         |         |         |
   66950 Pearfoot Motor Co. L      265.11  |             |         |    133.00 |     29.50 |         |         |         |         |  102.61 |
                                           |             |         |           |           |         |         |         |         |         |
   67000 Electricity               591.66  |             |         |           |    591.66 |         |         |         |         |         |
                                           |             |         |           |           |         |         |         |         |         |
                                           |             |         |           |           |         |         |         |         |         |
No. of accounts     9           58,980.92  |             |         | 39,396.38 | 19,481.93 |         |         |         |         |  102.61 |
                                           |             |         |           |           |         |         |         |         |         |
                       Percentages         |     0.00%   |  0.00%  |    66.80% |    33.03% |   0.00% |   0.00% |   0.00% |   0.00% |   0.17% |
                                           |             |         |           |           |         |         |         |         |         |
Account No   Short Name             Total  | Unallocated |  Later  |   Jul 88  |   Jun 88  |  May 88 |  Apr 88 |  Mar 88 | Earlier |   Held  |Comment
```

```
 9 Feb 82   9:49    The Honest Trading Company Ltd.          CUSTOMER ENQUIRY   RP                   ACC020    Page No 0002

LN A  B/F Y Akey 0 Req Y Arc N Pri ) 0 St A Sp 0 Single Accounts
O/P P PF 00 Pri )0

Account No.     56900    Feet Righten Process Engineering Ltd.,                            order terms as schedl        Page 001

Teesfield Works                 Thornaby-on-Tees
STOCKTON                        Cleveland                 TS20  2AQ      Telephone 0642 61979         Telex 33585

Turnover :-                     39,034.95          Current Balance :-                       21,745.39

Document  Document                                                          Payable      Balance    S  Pay
  Date    Reference Typ Remarks                    Value       Balance      Balance    (Incl Req)  T  No. Comments

  1.10.79 09.   9850 Fwd [illegible] - Advice [illegible]     2,500.00     2,500.00                   2,500.00  P   1
 10.10.79 09.   9861 Fwd [illegible] - Advice [illegible]       600.00     3,100.00                   3,100.00  P   2
 20.10.79 09.   9895 Fwd [illegible] - Advice [illegible]     1,000.00     4,100.00                   4,100.00  P   2
 19.11.79 09.   9923 Fwd [illegible] - Advice [illegible]     3,900.00     8,000.00                   8,000.00  P   2
 30.11.79 09.   9958 Fwd [illegible] - Advice [illegible]       723.50     8,723.50                   8,723.50  P   2
 10.12.79 09.   9980 Fwd [illegible] - Advice [illegible]     2,000.00    10,723.50                  10,723.50  P   2
  5.01.80 70.      2 Chq                       2,500.00-    8,223.50                   8,223.50  P   1
 10.01.80 32.   1000 Inv [illegible]                 542.57     8,766.07                   8,766.07  P   3
  5.02.80 70.      7 Chq                       8,223.50-      542.57                     542.57  P   2
  6.02.80 32.   1010 Inv [illegible]              12,465.30    13,007.95    12,465.30     13,007.95  S       6 May-Promises will be on
                                                                                                         next cheque.
                                                                                                         6 June-Regrets missed cheque
                                                                                                         in error. Will pay next week.
                                                                                                         20 June-Next week again.
                                                                                                         6 July-Not on Cheque. Refer
                                                                                                         to Mr Chapman. CONTINUE ACCT.
 13.02.80 32.   1017 Inv [illegible]              10,106.91    23,114.86                  23,114.86  P   4
  5.03.80 70.     12 Chq                         542.57-   22,572.29                  22,572.29  P   3
 15.04.80 32.   1037 Inv [illegible]               4,360.80    26,933.09                  26,933.09  P   5
 18.04.80 32.   1038 Inv [illegible]               2,279.28    29,212.37    14,744.66     29,212.37  S
  6.05.80 70.     21 Chq                      10,106.91-   19,105.46                  19,105.46  P   4
 10.05.80 32.   1043 Inv [illegible]               2,691.13    21,796.59    17,435.79     21,796.59  S
  5.06.80 70.     25 Chq                       4,360.80-   17,435.79                  17,435.79  P   5
 10.06.80 32.   1049 Inv [illegible]               2,947.76    20,383.55    20,383.55     20,383.55  S
  5.07.80 70.     27 Chq                       2,279.28-   18,104.27    18,104.27     18,104.27  S
 15.07.80 32.   1058 Inv [illegible]               3,641.12    21,745.39    21,745.39     21,745.39  S
```



```
21/03/83   11:00          The Honest Trading Company Ltd.                ACC020
                          Purchase/Sales Ledger Reports            RP

       Select accounts by Account No. (0) or Sort key (1) 0    Ledger S

    Account No. range:- 1 [          ] 2 [          ]  (P)aid/(N)ot Paid/(A)ll A

Dates:-     Type   Incl. B/F     From          To        With credits upto
             0       0       [        ]    [01.01.83]       [        ]

          Include register (Y or N) Y      Include archive (Y or N) N

      Current Balance between [5000.00   ] and [0.00      ]    Priority 0 0

    Document Type [ ]   Document Nos. from [      ] to [      ]    Special req. 0

    Enter individual accts/sort keys for selection, or for exclusion from range
       A [          ] B [          ] C [          ] D [          ] E [          ]
       F [          ] G [          ] H [          ] I [          ] J [          ]
       K [          ] L [          ] M [          ] N [          ] O [          ]
       P [          ] Q [          ] R [          ] S [          ] T [          ]

f1-payments list:f6-help:f8-exit
```



```
21/03/83   11:02          The Honest Trading Company Ltd.                ACC020
                          Purchase/Sales Ledger Reports            RP

                  Number of Accounts selected              [        13]

                               Posted            Registered          Archived
      Count of details:-   [        138]      [          4]      [          0]
      Value of details:-   [  59,620.86]      [     406.32]      [       0.00]
  Value brought forward:-  [       0.00]      [       0.00]

      Output device (T - terminal)        Print Format              Printer
                    (P - printer) P       (Press f6 for HELP) 03    [ LP3]

Only select accounts with an account priority       0 0
Only select accounts with selected balance between [100.00   ] and [0.00      ]

Type in selection & ENTER:f6-help:f7-return:f8-exit
```

Remittance Advices

Remittance advices may be printed at any time for any account and may contain any entry which has not been notified to the system as paid.

A typical instruction, illustrated opposite, could be:-

Print remittance advices for all accounts including invoices up to 31st January and any credits up to date.

This instruction could be varied by specifying the balance of the account, or the balance of the remittance advice which would be printed (ie pay everyone with a balance less than £100). It can also be varied by the users payment priority (pay only those suppliers with top priority) or by suppliers terms (pay overdue accounts only). If all else fails the user may examine a suppliers accounts and select one by one which items are to be shown on the remittance advice.

The printing of a remittance advice does not cause the system to think that the account has been paid. If open item remittance advices are printed all 'unpaid' items are available for selection. If brought forward then only those items added since the last real remittance advice was printed are available (dummy or draft remittances can be printed without changing this brought forward status).

Brought forward or open item are print choices and do not affect the information held by the system.

Remittance advices can be printed separately from cheques or an integral cheque/remittance advice can be produced.

Payments

Payments may be made by manually produced cheques entered into the system or by Cheques, Credit Transfers or BACS produced by the system.

Where cheques are made out in sequence but signed and released at a later date they may be placed on a register and 'entered' by releasing them within the system.

Where payments are covered by a remittance advice this can be communicated to the system. When the cheque is entered, it and all the items on the remittance, are marked as paid and will not appear on future remittance advices. If an attempt is made to enter a second cheque paying an invoice, the system reports the error. The cheque can however be entered as not paying anything and is treated as being on 'account'.

Optionally a proposed list of payments can be interactively assembled and manipulated on the VDU screen. A target date can be entered and, using the payment terms on the supplier file, the value of discounts which can be taken is calculated and listed. Once a Proposed Payments List is agreed upon it can be passed to the program which prints Cheques, Credit Transfers and BACS tapes.

The Honest Trading Company Ltd.
42 High Street
Outskirts
LEEDS
Yorkshire
LS1 3XX

Telephone 0532-431256

Telex 123456

Pearfoot Motor Co. Ltd.
Drivewell Garage
56 Seaton Drive
CARLTON
Notts
NT37 3BH

REMITTANCE
ADVICE

Account No. 66950

Date 9 Dec 81

Date	Our Ref		Type	Your Ref./Remarks	Debit	Credit	Balance
14 Jun 80	20	140	Inv	AG76995		29.50	29.50
23 Jul 80	20	144	Inv	AG77787		133.00	162.50
14 Aug 80	20	148	Inv	AG77931		77.99	240.49

This is a sample of a two line message to send to people.
Happy Christmas. Please send us money now!!!!!!!!!!!!!!!!!!!!

The Honest Trading Company Ltd.
42 High Street
Outskirts
LEEDS
Yorkshire
LS1 3XX

Pearfoot Motor Co. Ltd.
Drivewell Garage
56 Seaton Drive
CARLTON
Notts
NT37 3BH

66950

9 Dec 81

R15

Date	Our Ref.		£	Remarks
14 Jun 80	20	140	29.50	
23 Jul 80	20	144	133.00	
14 Aug 80	20	148	77.99	

£240.49

3. SALES LEDGER

As with the purchase ledger the sales ledger system allows staff to maintain firm control over the ledger and respond to the day to day pressures.

Customer File

The customer file contains the same comprehensive information as the supplier file. The same alpha sort key system is used so that it is possible to find a customer's account easily. For example, if the user sells to the Central Electricity Generating Board the account could be found by its account number or 'CEGB' or 'C.E.G.B.' or 'Central' or 'Electric' or 'Generati' or 'Board'.

Enquiries/Reports

As with Purchase Ledger there are a number of formats for enquiries which show information on a VDU or produce a printed report. These can show the detailed position of one or more accounts in a ledger card format or can show summarised or total information. Reports can be produced to show the age of debt either by invoice date or by due date. The main VDU enquiry is designed to give staff simple and rapid access to key detail information. Comments may be added, changed and deleted.

Sales Invoice Register

It is possible to maintain a register of sales invoices should this be necessary. This feature will be of use to organisations who have an invoice/despatch etc set. In these cases sets are issued in sequential order at time of despatch, or even at time of order, and are entered into the ledger out of sequence. This can cause problems and make it difficult to control and account for sets. The sales invoice register overcomes these problems by allowing a memorandum entry to be made when the set is allocated.

Statements

Customer's statements may be printed at any time for any account or accounts. Output may be in either open item format or brought forward format. Provision is made for the customer's order number and/or a comment to be shown on each entry.

Allocation/Mark-off

Receipts from customers are entered into the system and may be used to allocate or mark off items on the customer's account. The entry of receipts and their allocation allow opportunities to record discounts, part payments, payments on account, over and under payments and write offs and adjustments. For accounts with large numbers of open items allocation options are provided to automatically mark off all items on the account with the exception of certain documents or all items between two dates with the exception of certain documents. For those customers paying on statement or the balance of the account further automatic mark off options are available.

The Honest Trading Company Ltd.
42 High Street
Outskirts
LEEDS
Yorkshire
LS1 3XX

Telephone 0532-431256
Telex 123456

The Office Manager
Gilland of Teeside
Structural Engineers
Erection House
Merrick Trading Estate
Frame Road
STOCKTON-ON-TEES
Cleveland
TS18 2NH

STATEMENT

Account No. 56758
Date 31 Mar 82

order terms as sched2

Date	Our Ref.	Type	Your Ref./Remarks	Debit	Credit	Balance
12 Apr 80	32 1033	Inv	4383	1,468.14		1,468.14
15 May 80	32 1044	Inv	4328	4,331.37		5,799.51
10 Jun 80	32 1046	Inv	4335	813.40		6,612.91
10 Jun 80	32 1047	Inv	4335	2,099.58		8,712.49
10 Jun 80	32 1048	Inv	4335	573.42		9,285.91
5 Jul 80	70 28	Chq			5,799.51	3,486.40

This is a two line message which appears on this print run only.

The Honest Trading Company Ltd.
42 High Street
Outskirts
LEEDS
Yorkshire
LS1 3XX

The Office Manager
Gilland of Teeside
Structural Engineers
Erection House
Merrick Trading Estate
Frame Road
STOCKTON-ON-TEES 56758
Cleveland
TS18 2NH 31 Mar 82

S9

Date	Our Ref.	£	Remarks
12 Apr 80	32 1033	1,468.14	
15 May 80	32 1044	4,331.37	
10 Jun 80	32 1046	813.40	
10 Jun 80	32 1047	2,099.58	
10 Jun 80	32 1048	573.42	
5 Jul 80	70 28	5,799.51-	

3,486.40

```
26 Feb 81  13:14     The Honest Trading Company Ltd.          CUSTOMER ENQUIRY   PKT                  ACC020    Page No 0022

All        Accounts with a balance between  999999.99 and 9999999.99 and a priority  >0  chosen by Account  Including Register
All     Entries chosen from Earliest  to   Latest    Including Reverse Iupto  Latest

Account No.       57750    Stenia Metal Pressings                                  Order terms as sched 3         Page 001

Ljosnan
STAVANGER                     Norway                              Telephone 010 47 45 53428      Telex 6680428

  Turnover :-          36,769.06       Current Balance :-                                        0.00

                                                     Sterling                  Norwegian Kroner
Document Document                                              Payable    Balance                                 Payable     Balance         S Pay
  Date   Reference Typ Remarks                 Value    Balance   Balance  (Incl Reg)      Value     Balance     Balance   (Incl Reg)        T No. Comments

10.07.79 09.  0124 Fwd 1584 Oslo             2,164.71   2,164.71            2,164.71   27,416.50   27,416.50              27,416.50       P  1
30.09.79 09.  9120 Fwd 1780 Bergen           1,057.32   3,222.03            3,222.03   12,687.90   40,104.40              40,104.40       P  1
10.11.79 09.  9859 Fwd 2023 Oslo             7,952.00  11,174.03           10,174.03   95,433.50  135,537.90             135,537.90       P  2
31.12.79 09.   136 Fwd Interest                 60.70  11,235.61           11,235.61      717.50  136,255.40             136,255.40       P  2
16.01.80 72.   100 Clr                       3,645.85-  7,589.76            7,589.76   40,104.40-  96,151.00              96,151.00       P  1
16.01.80 72.   100 Clr Charge                   16.50-  7,573.26            7,573.26        0.00   96,151.00              96,151.00       P  1
16.01.80 72.   100 Clr ConvDf                  440.32   8,013.58            8,003.58        0.00   96,151.00              96,151.00       P  1
25.01.80 30.   101 Inv 2200 Oslo             5,647.06  13,660.64           13,660.64   72,000.00  168,151.00             168,151.00       P  3
 6.02.80 90.     2 Jnl Error on Conversion Diff.  1.00  13,661.64         13,661.64        0.00  168,151.00             168,151.00       P  2
 8.02.80 72.   102 Clr                       8,001.58-  5,660.06            5,660.06   96,151.00-  72,000.00              72,000.00       P  2
 8.02.80 72.   102 Clr Charge                   10.00-  5,650.06            5,650.06        0.00   72,000.00              72,000.00       P  2
 8.02.80 72.   102 Clr ConvDf                    3.00-  5,647.06            5,647.06        0.00   72,000.00              72,000.00       P  2
 8.03.80 72.   104 Clr                       5,602.50-     44.56               44.56   72,000.00-       0.00                   0.00       P  3
 8.03.80 72.   104 Clr Charge                   15.80-     28.76               28.76        0.00        0.00                   0.00       P  3
 8.03.80 72.   104 Clr ConvDf                   28.76-      0.00                0.00        0.00        0.00                   0.00       P  3
26.03.80 30.   105 Inv 2359 Oslo            10,000.00  10,000.00           10,000.00  127,500.00  127,500.00             127,500.00       P  4
15.04.80 30.   109 Inv ST9876 Oslo          10,450.00  20,450.00           20,450.00  133,237.50  260,737.50             260,737.50       P  5
20.04.80 72.   106 Clr                       9,872.00- 10,578.00           10,578.00  127,500.00- 133,237.50             133,237.50       P  4
20.04.80 72.   106 Clr Charge                   18.00- 10,560.00           10,560.00        0.00  133,237.50             133,237.50       P  4
20.04.80 72.   106 Clr ConvDf                  110.00- 10,450.00           10,450.00        0.00  133,237.50             133,237.50       P  4
21.05.80 30.   114 Inv St.00043-Bergen (II) 10,672.00  21,122.00           21,122.00  136,068.00  269,305.50             269,305.50       P  6
24.05.80 72.   109 Clr                      10,405.00- 10,717.00           10,717.00  133,237.50- 136,068.00             136,068.00       P  5
24.05.80 72.   109 Clr Charge                   15.75- 10,701.25           10,701.25        0.00  136,068.00             136,068.00       P  5
24.05.80 72.   109 Clr ConvDf                   29.25- 10,672.00           10,672.00        0.00  136,068.00             136,068.00       P  5
20.06.80 72.   111 Clr                      10,571.40-    100.60              100.60  136,068.00-       0.00                   0.00       P  6
20.06.80 72.   111 Clr Charge                   14.00-     86.60               86.60        0.00        0.00                   0.00       P  6
20.06.80 72.   111 Clr ConvDf                   97.60-     11.00-              11.00-       0.00        0.00                   0.00       P  6
21.06.80 90.     7 Jnl Adjust error in Conv. Diff.  11.00    0.00              0.00        0.00        0.00                   0.00       P  6
```

4. NOMINAL LEDGER

IAS/3000 is based on a nominal ledger and all entries, invoices, journals, cash etc take the form of entries to the nominal ledger. The ledger may have as many accounts as necessary to cover the analysis requirements of the various accounting departments and indeed inventory control departments.

Nominal Detail File

The nominal detail file is the largest file in an IAS/3000 database. Every entry in the books is recorded in it. Typically each invoice processed will make 4 or 5 entries in this file; a purchase or sales ledger control account entry, a VAT entry and 2/3 goods entries. Each entry includes the following information:-

Company number (01-99)
Nominal account (10 characters unstructured)
Ledger number (01-99 Sales & Purchase, EA-EZ Expense)
Sub-account (10 characters unstructured)
Document type (2 characters)
Document number (6 digits)
Value (16 digits plus 2 decimal places)
Description (30 characters)
Document date
Input date
Accounting year (01-99)
Accounting period (18 periods)
Quantity fields x 2 (15 digits plus 3 decimal places)
Status (P = paid, A = authorised, H = held etc)

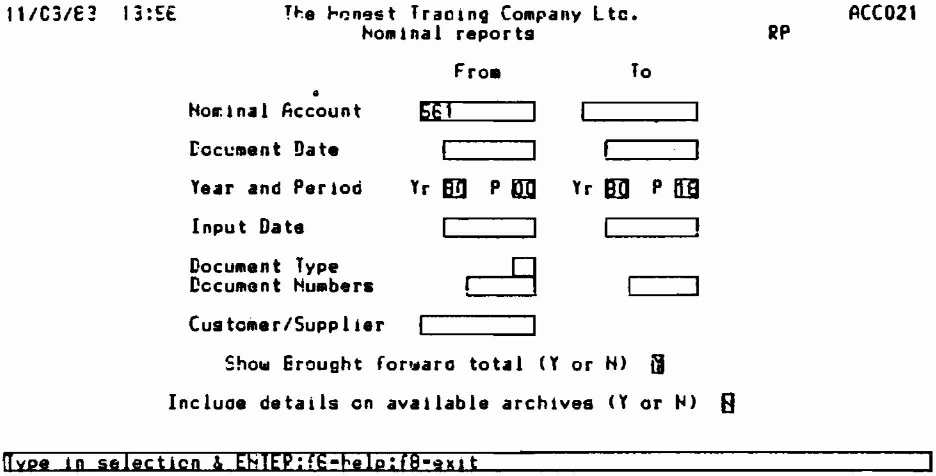
Nominal Ledger Sub-analysis

Any nominal account may be subanalysed using the sub account field; this is the key to the integratedness of IAS/3000.

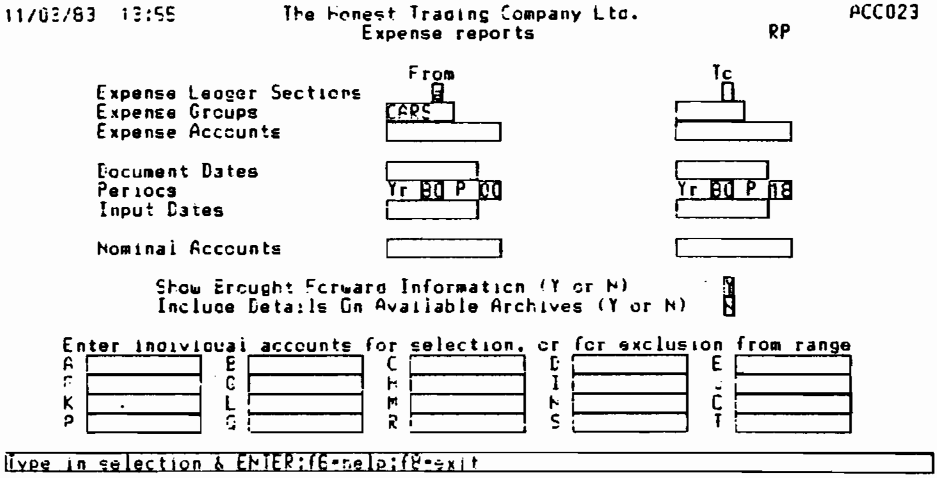
Up to 99 nominal accounts can be set up as special Sales or Purchase Ledger 'control' accounts. In these special cases the ledger number is 01-99 and the sub account is either a customer or supplier number. It can now be seen that Sales and Purchase Ledgers do not exist as separate entities; they are simply subdivisions of the Nominal Ledger.

Up to 26 other ledgers may be set up (EA-EZ) containing sub accounts which can be associated with any nominal account thus allowing a true two dimensional analysis between sub account and nominal account. Typical uses of this powerful feature might be to sub analyse:-

Vehicle expenses into car registration numbers
Telephone expenses into telephone numbers
Inventory into part numbers
Fixed assets into asset numbers
Job costs into job numbers
Contract costs into contract numbers



```
11/03/83  13:56        The Honest Trading Company Ltd.                    ACC021
                                   Nominal reports                     RP

                                             From              To

                 Nominal Account           561

                 Document Date

                 Year and Period          Yr 80  P 00      Yr 80  P 18

                 Input Date

                 Document Type
                 Document Numbers

                 Customer/Supplier

                    Show Brought forward total (Y or N)  Y

                 Include details on available archives (Y or N)  N

Type in selection & ENTER:f6-help:f8-exit
```



```
11/03/83  13:55        The Honest Trading Company Ltd.                    ACC023
                                   Expense reports                      RP

                                    From                        To
          Expense Ledger Sections      B
          Expense Groups            CARS
          Expense Accounts

          Document Dates
          Periods                   Yr 80 P 00                Yr 80 P 18
          Input Dates

          Nominal Accounts

             Show Brought Forward Information (Y or N)            Y
             Include Details On Available Archives (Y or N)       N

       Enter individual accounts for selection, or for exclusion from range
       A            B            C            D            E
       F            G            H            I            J
       K            L            M            N            O
       P            Q            R            S            T
Type in selection & ENTER:f6-help:f8-exit
```

Quantity fields

Usage of the two quantity fields allows the total integration of inventory and book keeping systems. The quantity fields have 3 decimal places to allow for units such as tonnes.

Nominal & Sub account Enquiries/Reports

All reports are available on demand for one account, a range of accounts or all accounts.

Selection may be by date of document, date of input, accounting period and type of document.

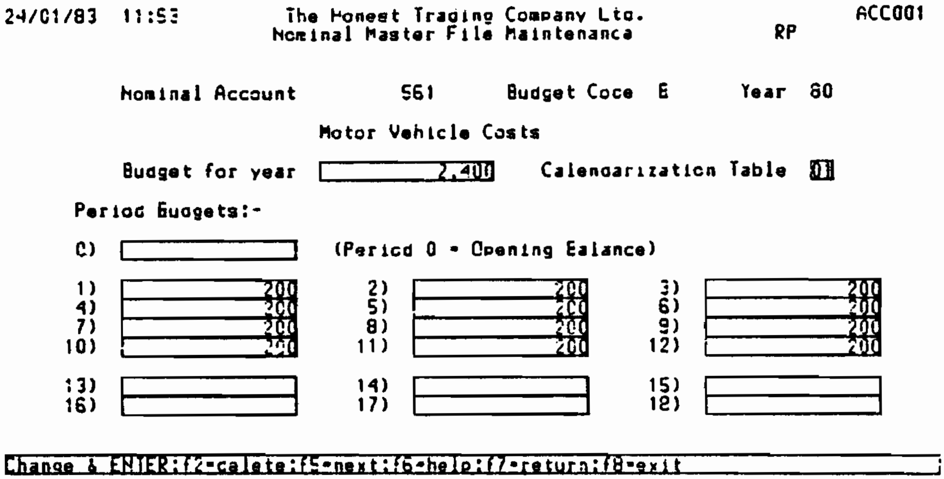
The selected entries may be sorted and summarised at various levels.

Output may be to VDU or printer.

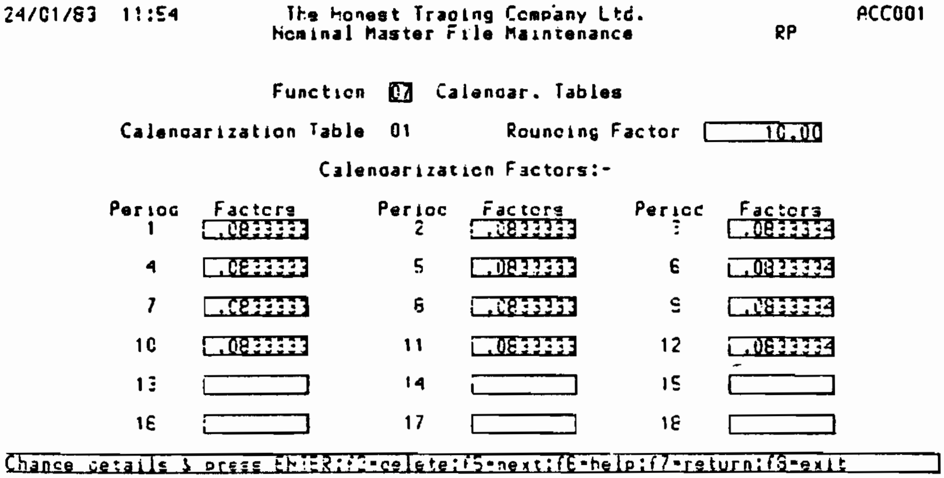
Selection screens are shown opposite and typical reports overleaf. All entries posted to nominal account 561 Motor Vehicle Costs have been selected on the first screen. All expense accounts in the group called CARS on the second screen.

```
 7 Jan 83  13:45      The Honest Trading Company Ltd.            Nominal Account Report          Mr Rodney Potts       Page No.0001

Acct 1 561 Acct 2 561 Per 1 80 00 2 80 10 B/Fwd Y Acc N
Dev P Level 0 ValForm 0 XQVal Q Seq 2 Comp Y Printer LP31

Account Number      561  Motor Vehicle Costs            Current Balance       502.11   Department     10

(------DOCUMENT------)                                          Per-  Input  Init Expense                                 (------XREF------)
 Date     Reference  Description             Value        Balance  iod    Date  ials Account  Title                        Account   Sort Key

                     Brought Forward          0.00          0.00                                                                                     0.000        0.000

                          Total period 0000   0.00                                                                                                   0.000        0.000

20.01.80 20 Inv  107 Repairs                 39.00         39.00  0002 22.02.81 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    66950 PEARFOOT   0.000        0.000
31.01.80 20 Inv  111 Repair                  39.54         78.54  0002 22.02.81 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    66950 PEARFOOT   0.000        0.000
31.01.80 20 Inv  111 Parts                   23.26        101.80  0002 22.02.81 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    66950 PEARFOOT   0.000        0.000
24.02.80 20 Inv  118 Service                 68.56        170.36  0002 22.02.81 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    66950 PEARFOOT   0.000        0.000

                          Total period 0002 170.36                                                                                                   0.000        0.000

29.02.80 20 Inv  120 Repairs                 20.10         20.10  0003 22.02.81 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    66950 PEARFOOT   0.000        0.000
29.02.80 20 Inv  120 Parts                   10.00         30.10  0003 22.02.81 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    66950 PEARFOOT   0.000        0.000
29.02.80 20 Inv  121 Repairs                 59.20         89.30  0003 22.02.81 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    66950 PEARFOOT   0.000        0.000
29.02.80 20 Inv  120 Service                 62.20        151.50  0003 22.02.81 BP  NYG446V NYG446V Austin Maxi Mr CC Sellers    66950 PEARFOOT   0.000        0.000
29.02.80 20 Inv  120 Parts                   31.90        183.40  0003 22.02.81 BP  NYG446V NYG446V Austin Maxi Mr CC Sellers    66950 PEARFOOT   0.000        0.000

                          Total period 0003 183.40                                                                                                   0.000        0.000

16.04.80 20 Inv  130 Service                 89.23         89.23  0005 23.02.81 BP  NYG446V NYG446V Austin Maxi Mr CC Sellers    66950 PEARFOOT   0.000        0.000

                          Total period 0005  89.23                                                                                                   0.000        0.000

 6.01.83 30 Inv  233 hgfhgfhgfhfh           100.00-       100.00- 0006  6.01.83 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    56600 VICAR      0.000        0.000
 6.01.83 30 Inv  233 hgfhgfhgfhfh           750.00-       850.00- 0006  6.01.83 BP  NYG446V NYG446V Austin Maxi Mr CC Sellers    56600 VICAR      0.000        0.000
 7.01.83 20 Inv  154 Repairs to headlamp    100.00        750.00- 0006  7.01.83 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    66950 PEARFOOT   0.000        0.000
 7.01.83 20 Inv  154 Repairs to headlamp    600.00        150.00- 0006  7.01.83 BP  NYG446V NYG446V Austin Maxi Mr CC Sellers    66950 PEARFOOT   0.000        0.000

                          Total period 0006 150.00-                                                                                                  0.000        0.000

23.07.80 20 Inv  144 Service                115.65        115.65  0008 23.02.81 BP  NYG446V NYG446V Austin Maxi Mr CC Sellers    66950 PEARFOOT   0.000        0.000
14.08.80 20 Inv  148 Repair                  45.41        161.06  0008 23.02.81 BP  NYG446V NYG446V Austin Maxi Mr CC Sellers    66950 PEARFOOT   0.000        0.000
14.08.80 20 Inv  148 Parts                   22.41        183.47  0008 23.02.81 BP  NYG446V NYG446V Austin Maxi Mr CC Sellers    66950 PEARFOOT   0.000        0.000

                          Total period 0008 183.47                                                                                                   0.000        0.000

14.06.80 20 Inv  140 Headlamp repair         25.65         25.65  0009 23.02.81 BP  CUB216V CUB216V Jaguar XJ12 Mr JV Chapman    66950 PEARFOOT   0.000        0.000

                          Total period 0009  25.65                                                                                                   0.000        0.000

                      Total Nominal Account 502.11                                                                                                   0.000        0.000
```

```
11 Mar 83   13:56       The Honest Trading Company Ltd.                 Expense Account Report          Mr Rodney Potts       Page No.0001

Sect 1 A Sect 2 A Grp 1 CARS Grp 2 CARS Per 1 80 00 2 80 19 B/Fwd Y Arc N
Dev P Level D Valform 0 Seq 0 Comp Y Printer LP31

Ledger Section  A      Expense Analysis

Account Number      CUB216V  CUB216V Jaguar X12 Mr JV-Chapman           Current Balance            285.31    Expense Group    CARS

(======DOCUMENT======)                                                Per-  Input  Init (======Cross-Reference======) (==============N O M I N A L==============)
 Date     Reference  Description                       Value         Balance iod   Date   ials  Account Title                 Account Title                                  Quantity    Quantity2

20.01.80 20 Inv    107 Repairs                          39.00          39.00 0002 22.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
31.01.80 20 Inv    111 Repair                           39.54          78.54 0002 22.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
31.01.80 20 Inv    111 Parts                            23.26         101.80 0002 22.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
24.02.80 20 Inv    110 Service                          68.56         170.36 0002 22.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
29.02.80 20 Inv    120 Repairs                          20.10         190.46 0003 22.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
29.02.80 20 Inv    120 Parts                            10.00         200.46 0003 22.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
29.02.80 20 Inv    121 Repairs                          59.20         259.66 0003 22.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
14.06.80 20 Inv    140 Headlamp repair                  25.65         285.31 0009 23.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs

                Total Expense Acct    CUB216V         285.31

Account Number      NYG446V  NYG446V Austin Maxi Mr CC Sellers          Current Balance            366.80    Expense Group    CARS

29.02.80 20 Inv    120 Service                          62.20          62.20 0003 22.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
29.02.80 20 Inv    120 Parts                            31.90          94.10 0003 22.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
16.04.80 20 Inv    130 Service                          89.23         183.33 0005 23.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
23.07.80 20 Inv    144 Service                         115.65         298.98 0008 23.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
14.08.80 20 Inv    140 Repair                           45.41         344.39 0008 23.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs
14.08.80 20 Inv    140 Parts                            22.41         366.80 0008 23.02.81 RP   66950 Pearfoot Motor Co. L    561 Motor Vehicle Costs

                Total Expense Acct    NYG446V         366.80

                Total Expense Section A               652.11
```



```
24/01/83  11:53          The Honest Trading Company Ltd.                    ACC001
                        Nominal Master File Maintenance            RP

           Nominal Account          561      Budget Code  E      Year  80

                                  Motor Vehicle Costs

           Budget for year  [      2,400]    Calendarization Table [01]

         Period Budgets:-

         0)  [          ]   (Period 0 = Opening Balance)

         1)  [       200]    2)  [       200]    3)  [       200]
         4)  [       200]    5)  [       200]    6)  [       200]
         7)  [       200]    8)  [       200]    9)  [       200]
        10)  [       200]   11)  [       200]   12)  [       200]

        13)  [          ]   14)  [          ]   15)  [          ]
        16)  [          ]   17)  [          ]   18)  [          ]

Change & ENTER:f2=delete:f5=next:f6=help:f7=return:f8=exit
```



```
24/01/83  11:54          The Honest Trading Company Ltd.                    ACC001
                        Nominal Master File Maintenance            RP

                          Function [07] Calendar. Tables

           Calendarization Table  01        Rounding Factor [    10.00]

                              Calendarization Factors:-

          Period    Factors        Period    Factors        Period    Factors
             1     [.083333]          2     [.083333]          3     [.083334]

             4     [.083333]          5     [.083333]          6     [.083334]

             7     [.083333]          8     [.083333]          9     [.083334]

            10     [.083333]         11     [.083333]         12     [.083334]

            13     [       ]         14     [       ]         15     [       ]

            16     [       ]         17     [       ]         18     [       ]

Change details & press ENTER:f2=delete:f5=next:f6=help:f7=return:f8=exit
```

The Nominal Master File

The nominal master file includes the following information:-

- Company number (01-99)
- Nominal account (10 alphanumeric)
- Period balances (18 periods)
- Opening balance
- Year End adjustment balances
- Current balance
- Year number (01-99)
- Budget code (A-Z)
- Department code (6 alphanumeric)
- Report lines (Rlines) (6 sets)

For each nominal account 26 sets of balances (Budget code A-Z) can be maintained for each of 99 years. All of the balances are available to be reported on in final formatted accounts using the **IAS/3000 Financial Reporting System.**

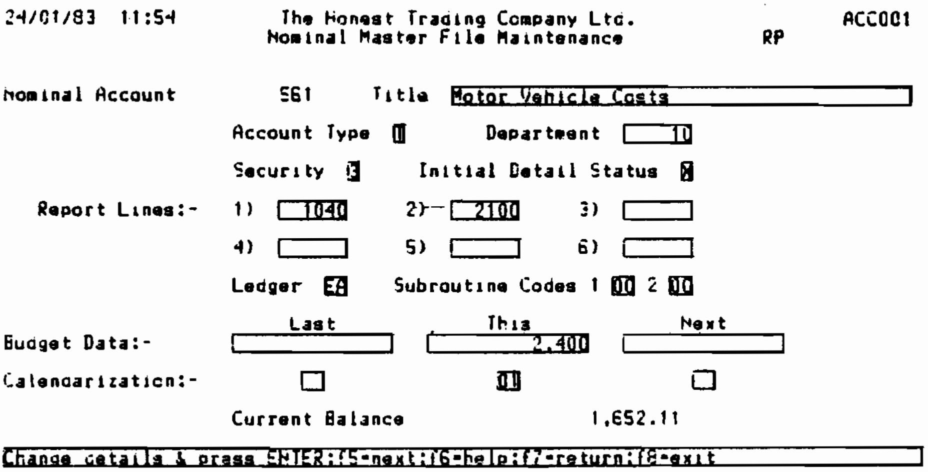
The 'A' set of balances is reserved for actuals and the 'X' set for archive summary records. The other 24 sets are typically used as 'B' for budgets, 'E' for estimates, 'F' for forecasts etc.

Actuals

The 'A' actual balances can only be accessed by making real entries in the books. When an invoice is posted it typically puts a number of entries in the nominal detail file and at the same time updates the appropriate nominal master period balance and current balance.

Budgets

The 'B', 'E', 'F' etc period balances can be easily manipulated manually. Budgets can be entered as an annual figure and then calendarised or spread over the up to 18 period in the year using one of 99 tables of factors. Alternatively individual period figures can be entered.



```
24/01/83   11:54          The Honest Trading Company Ltd.                    ACC001
                         Nominal Master File Maintenance            RP

Nominal Account          561     Title  Motor Vehicle Costs

                  Account Type 0        Department     10

                  Security  3     Initial Detail Status  Y

   Report Lines:- 1)   1040     2)--  2100      3)

                  4)            5)              6)

                  Ledger EA    Subroutine Codes 1 00 2 00

                         Last            This             Next
Budget Data:-                              2,400

Calendarization:-         ☐               01               ☐

                  Current Balance             1,652.11

Change details & press ENTER:f5-next:f6-help:f7-return:f8-exit
```

5. FINANCIAL REPORTING SYSTEM

The 6 sets of Report Lines (RLINES) on the nominal master record provide the accountant with 6 different ways of merging or adding together nominal accounts. This makes it possible to meet the different reporting requirements of head office, company, directors, managers and the accountant.

The merging of accounts can be hierarchial but needn't be. Each nominal account can be merged into a different Rline in up to 6 different reports. In addition 2 trial balance report formats with a line per nominal account and subtotals at department level can be produced as standard.

Three programs in the IAS/3000 package together comprise the **Financial Reporting System**. They are fill-in-the-forms programs designed so the accountant can design his own reports.

The first program RLINES designs the top-to-bottom formatting and arithmetic of the report using the 6 sets of Rlines.

The second program REPDEF designs the headings, columns, left-to-right arithmetic across the report and the sources of data, budgets, actuals for this year, last year etc to go into the columns. It is possible to have any number of these report definitions (REPDEFs).

The third program REPORT actually prints the report and allows the combination of any REPDEF with any set of RLINES. Hence although there are only 6 ways of merging nominal accounts (plus the two trial balance structures) there are, however, an infinite number of report formats (combinations of RLINES and REPDEF).

The first page of a typical set of month end accounts produced by FRS is shown overleaf along the set of RLINES which produced them.

The Honest Trading Company Limited
Accounts for the Month of June 1980

Last Year			Current Month			Year To Date		
Month	Y.to Dte		Budget	Actual	Variance	Budget	Actual	Variance
		Sales:						
		Home Sales	25.140	13.275	(11,865)	150,840	185.629	34,789
		Cost of Sales	21,450	11.663	(9,787)	128,700	141,378	12,678
		Home Gr. Margin	3,690	1.612	(2,078)	22.140	44.251	22,111
		Export Sales	20,000		(20,000)	120,000	119.773	(227)
		Cost of Sales	16,670		(16,670)	100,000	109,440	9,440
		Export Gr. Margin	3,330		(3,330)	20,000	10,333	(9,667)
		Total Gross Margin	7,020	1,612	(5.488)	42,140	54,584	12.444
		Overheads						
		Employment Costs	3.350	2,991	(359)	20.100	14.667	(5.433)
		Expenses	1.650	2.287	637	11,950	9,964	(1,986)
		Depreciation	1,500	3,097	1,597	3,000	4,774	1,774
		Discounts						
		Total Overheads	6.500	8.375	1,875	35,050	29,405	(5,645)
		Net Profit	520	(6,763)	(7,283)	7,090	25,179	18.089
		Interest Received	2,000	3,150	1,150	2,000	3.150	1,150
		Interest Paid						
		Profit before Tax	2,520	(3.613)	(6,133)	9,090	28.329	19,239

```
11 Mar 83  15:03        The Honest Trading Company Ltd.       List of report line number definitions produced by RP      Page No.0001

Report Rep                           Pag  Spac  Lin  Rev  Zero Allow
 Line  Num Title                     ing   ing  ing  Val  Supp Data    Totalling

  2010  2  Sales:                     N    1 A   N    N     T    N

  2020  2     Home Sales              N    0 B   N    Y     T    Y      1 A

  2030  2     Cost of Sales           N    0 B   N    N     T    Y      1 A

  2040  2     Home Gr. Margin         N    1 A   N    Y     T    N      1 PC  2 A

  2050  2     Export Sales            N    0 B   N    Y     T    Y      1 A

  2060  2     Cost of Sales           N    0 B   N    N     T    Y      1 A

  2070  2     Export Gr. Margin       N    1 A   N    Y     T    N      1 PC  2 A

  2080  2  Total Gross Margin         N    2 A   N    Y     T    N      2 P

  2085  2  Overheads                  N    1 A   N    N     T    N

  2090  2  Employment Costs           N    0 B   N    N     T    Y      1 A

  2100  2  Expenses                   N    0 B   N    N     T    Y      1 A

  2110  2  Depreciation               N    0 B   N    N     T    Y      1 A

  2120  2  Discounts                  N    0 B   N    N     T    Y      1 A

  2125  2  Total Overheads            N    1 A   N    N     T    N      1 PC  2 A

  2130  2  Net Profit                 N    1 A   N    Y     T    N      2 P

  2140  2     Interest Received       N    0 B   N    Y     T    Y      2 A

  2150  2     Interest Paid           N    1 A   N    N     T    Y      2 A

  2160  2  Profit before Tax          N    0 B   N    Y     T    N      2 PC  4 A
```

COMPUTERIZED PERSONNEL SYSTEMS

Tony Ive

Manager Personnel Projects
Foster Wheeler Energy, Reading.

1. Background

There is a considerable amount of interest at the present time in computerised personnel systems and many DP Managers have requests outstanding from their Personnel Managers for advice as to what they should do in this area. However, it is a notoriously difficult area of system development work and many projects which have been undertaken in this field have become casualties. DP Managers who have prior experience of attempts to computerise in this area are therefore well aware of the special difficulties.

This paper is concerned with the development work that has been done over the last eight years at Foster Wheeler on computerised personnel systems. Foster Wheeler Energy Limited is a petrochemical contractor based in Reading employing around 2,000 people. The personnel systems which are in use have been developed on the HP3000 and made a dramatic difference to the role which the Personnel Department has been able to play.

The paper begins by analysing the problems which the Personnel Department faced and it is the belief of members of the Personnel Department that this analysis would not vary much from one company to another. The paper then goes on to outline the solutions which result from the computerised systems on the HP3000 and concludes with a summary of some key design considerations.

2. Problems

The problems of the Personnel Department at Foster Wheeler fall into three broad categories, recruitment, personnel records and the annual salary review. This breakdown of the problems is in itself significant because many computerised personnel systems fail to take account of the importance of recruitment as far as a Personnel Department is concerned, and also often overlook the special problems of carrying out the annual salary review.

In this breakdown of the problems, personnel records of course covers a wide area of activities which include routine salary administration, training records, a detailed inventory of skills and such things as absence and sickness. A detailed analysis of these problems was carried out in Foster Wheeler in 1976 and as a result of this, three elements were identified which impinged on each of these problems and were, in a sense, responsible for creating them. These elements were record maintenance, letter production and information management. This description is too broad to be of any use on its own and so to clarify the nature of the problems, each of them will be considered in turn with specific reference to these three elements.

2.1 Recruitment

The reason why recruitment causes Personnel Departments such severe problems is that the level of administrative work necessary to cope with a recruitment campaign fluctuates by a very large amount. In a small Personnel Department there will be some times when there are no vacancies to be filled at all and others when several have to be filled at the same time.

When a vacancy is to be filled, the first step is to place an advertisement in the paper. This may well generate several hundred replies and each of these replies is likely to result by the end of the campaign in on average three letters. Hence if several such campaigns are progressing at the same time, a very large volume of paperwork is generated. In larger companies the number of vacancies to be filled may never fall to zero but there will still undoubtedly be very great fluctuations in the level of activity in the recruitment section of the department as far as the basic administration tasks are concerned.

At Foster Wheeler the workload fluctuates by a factor of about ten and it is these fluctuations which are the essence of the problem. If the workload were high but constant it would be possible to deal with it simply by employing more staff. But since it is not constant there are some times when people are under-employed and others when the whole department is thrown into disarray by the need to reallocate people from their normal work to assist with recruitment and also by the need to employ temporary staff to help process the paperwork. Thus, letter production, mundane though it sounds, is the starting point of of the difficulties for a Personnel Department which is engaged in a recruitment campaign.

This problem leads directly into the second one: record maintenance. If everyone in the department is working flat out to write the required number of letters, it is virtually impossible for them to maintain adequate records of what they are doing at the same time. Again and again the decision has to be taken: is the priority to get these letters out or is the priority to update the index card system and the associated listings of applicants in each vacancy to reflect the up-to-date position. Under the pressures of the campaign the answer is always to send out the letters and to leave the updating of records until later. Naturally, what eventually happens is that the records do not get updated or are not updated completely and accurately. Applicants then begin to phone up to find out what the status of their application is and the Personnel Department is unable to find any record of it. At this point the recruitment process begins to break down.

The problem of record maintenance leads again to the third problem, that of information management. During a recruitment campaign, information is required at several levels of detail. Each recruiter needs to have a list for every vacancy for which he or she is recruiting showing all people who have applied and the dates when their application forms were received, when they were interviewed, when contracts of employment were sent to them, etc. Managers require summarised information for the vacancies which occur within their departments and top management are interested in overall statistics about the number of vacancies in the company, how they break down, how many people have applied, how many are being interviewed, how many have been sent offers of employment, how many have rejected them and why. These are the tasks which the Personnel Department is likely to find next to impossible:

impossible in the first place because the basic records required to answer such questions do not exist for reasons explained above; impossible in the second place because even if the records did exist it would take an unacceptably long time to go through them manually to produce information in all the different forms in which it is required. When the problems of recruitment administration are looked at closely in this way, it is easy to see why they cause such disruption in a manual system. But it is not easy for an outsider to appreciate these problems and Personnel Managers frequently find themselves in an embarrassing position, trying to explain to top management why the various sets of figures they have produced for recruitment statistics do not tie up, or why they are failing to perform such seemingly simple and mundane tasks satisfactorily. It is these pressures often that lead a Personnel Manager to feel he ought to be able to get some help from a computer.

2.2 Personnel Records

In personnel records it is clear that the element record maintenance is the key to the problem. How exactly does this manifest itself? Basically it is manifested in the fact that, because manual records are so difficult to manipulate and because Personnel Departments know that from time to time they will be required to manipulate them in certain ways, the information ends up being stored in a large number of different places: a personal file for each individual, a card index system arranged in alphabetical order to enable immediate access to be made to an individual's details, a card index system stored alphabetically within department, (because the majority of manpower breakdowns which a Personnel Department is likely to be asked to perform will approximate most closely to an initial sort by department), separate lists of institute membership, all courses and all languages spoken, certain medical records stored separately, certain information relating to salary surveys in which the company regularly participates kept separately again, and selected individuals in the department having the responsibility for maintaining lists of such things as people who qualify for BUPA membership or those with company cars.

This proliferation, duplication, triplication and quadruplication of information results not only in a large amount of unnecessary work but also and inevitably in inconsistencies. Someone who exists in the training records does not exist on the main personnel records or they are in a different department or their job title is different. None of the records is entirely accurate and when analyses have to be performed which cross the boundaries of the different record systems, insuperable difficulties arise.

Letter production is not such a serious problem in personnel records as it is in recruitment. Nevertheless it is a significant one. The majority of changes processed by a Personnel Records Officer will be changes in department, location, salary or job title and all of these require an amendment to the contract of employment. Again, the problem is similar to recruitment: the change is processed on the record system but the required documents are not produced until later. They may never be produced or, through occasional human error, they may contain different information from what is stored on the records.

The element which was referred to broadly as information management is the one which everyone immediately thinks of in connection with personnel records - everything from the simple lists which every Personnel Department has to produce on a weekly or monthly basis, such things as starters, leavers, salary increases, transfers, promotions, etc., through to the classic enquiries of the form, "Who have we got who is aged between 25 and 35, has at least five years experience with the company, is qualified to degree level, is currently based in the head office and has experience of working with dog kennels?" to the more complicated statistical analyses such as training board returns the input to salary surveys, and breaking down job functions in the company in a specified way showing how many people within each function have, for example, nought to two years service with the company, 2-5 years, 5-10, 10-15, 15-25, 25 and over, or performing the same analysis for years experience in the industry. These are the sort of enquiries which are a nightmare under a manual system.

2.3 Salary Reviews

It was mentioned earlier that the annual salary review requires separate consideration because of the special problems which it involves. Of course, not all companies have an annual salary review. Some operate a rolling review system. However, the essential problems are the same, and the element with which the problem of salary reviews begin is information management. For a number of months (in some organisations this has virtually become a number of years) discussion ranges extensively over the question of how the review should be carried out and how much it will cost. What would be the effect of having X% here, Y% there, and a special package somewhere else? What happens if we take ½% off all these? The number of permutations that can be and are considered is virtually infinite and even in companies which are highly autocratic and do not indulge in discussions with their employees a large amount of time is wasted by senior managers when they are trying to work out the cost of implementing a proposed policy. When agreement is finally reached, the Personnel Department is instantly confronted with two major problems: producing a very large number of letters and updating a very large number of personnel records. In many organisations agreement is not normally reached until after the date when the changes should have become effective, and even in those which avoid this special difficulty, it is common for the decision not to be taken until the very last possible moment.

This means that the Personnel Department are under pressure to carry out these two activities in a very short space of time. Both of them, however, are very time consuming, are entirely additional to the normal work of the department, and because of their confidential nature are not normally suitable areas in which to rely on temporary assistance. The supposed inefficiency of the Personnel Department in not carrying out this task in zero time is frequently a major contributory factor to their unpopularity.

In each of these areas - recruitment, personnel records and the carrying out of annual salary reviews - the Personnel Department is performing apparently mundane tasks which on closer examination are very much more difficult than they at first appear. In particular they suffer from an inability to manipulate information as quickly and as freely as they would like to and an inability to deal efficiently with areas in which severe fluctuations of workload are encountered. The net result of this is that a credibility problem exists for the Personnel Department which not only embarrasses the Personnel Manager but prevents him from getting the support he needs to carry out some of the more policy-orientated aspects of personnel work. It is likely to be these difficulties which are in the back of the Personnel Manager's mind when he approaches the DP Manager for advice. The stock reply, "Give me a user specification and I will tell you what to do", is extremely unhelpful. When someone asks for general advice, their requirements are not met by having an obligation placed upon them to write a specification. However, the DP Manager too is in a difficult position: how does he know what the Personnel Department do unless the Personnel Manager tells him, let alone what they would like to do?

3. Solutions

If the three elements of the problems we have examined are record maintenance, letter production and information management, then very glibly the solutions are:

1) do away with all manual records and store everything once only on a data base.

2) design the system around an integrated letter writing system (not word processing - the difference is explained below).

3) use a powerful report generator.

As with the problems, the best way to clarify the solutions is to deal with each of the areas in turn:-

3.1 Recruitment

The crux of the problem in recruitment administration is the fluctuations in the level of activity by a factor of about ten. How can such a problem be solved? The fluctuations are in the nature of the activity, therefore in themselves they cannot be removed.

The solution is to eliminate their effect by speeding up the administrative process to such an extent that they no longer have a significant effect. In other words, the method of handling the applications must be speeded up by a factor of about ten. How can this be done? In the first place, the workload can be halved by combining the process of letter writing with that of record maintenance. This is what was referred to earlier as integrated letter writing. The idea is that an indissoluble link is established between the updating of the data base and the generation of any associated letters or documents. The person operating the system sits in front of a formatted screen and when, for example, an entry is made to indicate that applicant number 1030 is to receive letter 3L (an interview letter) the cursor will be positioned at interview time and interview date and when carriage return is pressed the data base will be updated and the letter generated at one and the same time. Hence, where previously two tasks had to be performed, each requiring similar amounts of time, there is now only one task to be performed.

The process is further speeded up by comparison with a manual system by ensuring that the programme is sufficiently intelligent to make sensible assumptions about what is going on and so to enable corners to be cut. Almost every stage in the recruitment process will involve sending someone a letter, and reduction of recruitment process to a series of standard letters is therefore fundamental. If an applicant is called up and the letter to be sent is indicated, the system should be able to determine which additional fields need to be accessed and to avoid wasting the operator's time by going through other fields or by asking unnecessary questions. Further devices for saving time are such things as chaining facilities and a hold and terminate facility. A chaining facility is simply a means of getting the system to call up applicants' details in the correct order where this order is pre-defined instead of asking the operator to do it. This arises where a number of applicants are being processed together (the normal situation in a recruitment campaign). The terminate facility is the ability to tell the system that a particular vacancy has now been filled and so to cause it to send regret letters to all applicants who are still in an unresolved state. It may be necessary for the system to distinguish between applicants who have been in for interview and those who have not. A hold facility is similar to a terminate facility but is used where a delay occurs during the recruitment process and it is necessary to write to all applicants to tell them what is going on.

The difference between integrated letter writing and word processing is that word processing is primarily concerned with the reformatting of text, whereas integrated letter writing is conerned with the combination and manipulation of data from a variety of different sources. Word processing is ideal for tasks such as drafting a report - paragraphs can be swapped around, pieces can be taken out and pieces can be put in. However, a Personnel Department does not redraft its contract of employment every time it makes an offer to an applicant. The basic elements of the contract are determined partly by law and partly by the company's personnel policy. What the company does want to do is to combine the relevant contractual conditions with details of the applicant as stored on the data base and details of the conditions of employment in the place where he or she will be working, to carry out any necessary calculations on salary, and to prompt the operator with questions concerning any aspects of a contract of this sort which may be determined on an individual basis. This is a data processing operation rather than a word processing one. The kind of reports required for recruitment administration were mentioned earlier in this paper, and all of them should be perfectly possible with a powerful report generator.

3.2 Personnel Records

Solution of the problem of personnel records requires a similar approach. Record maintenance is tackled by means of easy-to-use formatted screens. Certain updates on these screens trigger certain other updates, for example, a change to a person's job title, salary, department or location on the main screen should also trigger an update to the employment history screen and should cause a letter to be produced.

There are some important design considerations, but these are dealt with in the next section. The main advantages of the personnel records system will derive from the provision of a powerful report generator. It is of course essential that members of the Personnel Department should be able to use this themselves. At Foster Wheeler, the Personnel Department has written about 100 standard reports using the report generator which are stored in EDITOR files within the system. The Personnel Records Officer and other members of the department all use the report generator on a regular basis to deal with ad hoc enquiries. The report generator should be able to manipulate any information which is stored within the security confines of the individual concerned in more or less any way that they want to.

3.3 Salary Reviews

The starting point of the problems on the annual salary review was the manipulation of information to ascertain the effect of a given policy. The way to solve this is to extract information relevant to a salary review from the data base into a work file and to provide an easy-to-use formatted screen which enables modifications to the work file to be entered. It should be possible to modify it on either an across-the-board or an individual basis and exceptions may have to be made by reference to job title or range of job titles, salary or range of salary.

The second part of the problem with salary reviews was how to generate all the letters and update all the personnel records when a policy decision had been made. Here again, as in the case of recruitment, dramatic advantages can be obtained from the sensible use of a computer. If a work file has been manipulated and the effects of a given policy calculated it makes sense to provide a single command which will update all the records and produce all the letters without any further intervention from the operator.

It may be sensible to provide two other facilities, one of which actually implements the salary review and the other of which (likely to be of relevance in a company which operates a rolling review) to set up budgets for the coming year. The actual update would cause proposed salary on the work file to be written to current salary on the data base, updating employment history and producing a letter at the same time. A budgeted update would simply cause proposed salary on the work file to be written to budgeted salary on the data base.

At Foster Wheeler it used to take about six weeks under the manual system to update the personnel records and produce the salary letters for 2,000 people (three weeks to produce the letters and three weeks to update the records). With the computerised system it would be possible to input the parameters for the change, update the records and report the letters to a file within half an hour. Printing the letters off on a daisy-wheel printer would take longer of course but it would still be possible to turn around the entire review in a single day.

The essence of this approach - combining data base updates with letter production and using a report generator for manipulation and extraction - solves one of the most serious problems of the Personnel Department : inconsistency between information generated from different sources.

4. Design Considerations

An attempt has been made to analyse the problems of a Personnel Department and to indicate the solutions which can be found through the use of a computer. It is now relevant to highlight several of the key points to be borne in mind when such a system is designed.

4.1 Extensive Systems and Procedural Analysis

It must be appreciated that a system of the kind which has been described in this paper has to be fully integrated into the administrative procedures of a Personnel Department. It is not something which can exist alongside current procedures. The work of everyone in the department will change as a result of the implementation of such a system and there are therefore very important training and organisational considerations. If a system is to integrate successfully into the administration of a Personnel Department it must replace all the existing manual systems and procedures. Members of the Personnel Department will not find it acceptable if they have to continue their manual records alongside the computer system,but to do this requires an unusually extensive degree of systems and procedural analysis.

It is only reasonable to mention at this stage that some of the problem areas referred to in this paper, when examined in detail, will be found to be substantially more complex than might be thought. Computerised personnel systems are not simply a matter of fitting a large amount of information into a computer system and then getting it out again in a different form. A recruitment system, for example, is more similar to a production system than to an ordinary information system. It has to store information about the stages through which applicants pass. The difficulty of designing such a system arises from the fact that the paths which applicants follow in the course of being recruited vary enormously. To take a simple example, a large amount of the space required on the record of an applicant is taken up by information relating to the interview and the offer of employment; the question must obviously be asked, " What is the maximum number of interviews which the applicant might have and what is the maximum number of offers?" Few Personnel Managers will be keen to commit themselves on this point, but whatever answer is given it is certain soon to emerge that an enormous amount of space will be taken up by repeats

The answer to the second question is that the kind of flexibility which is really required is the kind which is supplied bv the provision of a powerful report generator, and speed Is not of overriding importance here. The Personnel Department is moving from a position where it either cannot handle ad hoc enquiries satisfactorily at all or takes a very long time to do so. The important achievement of the system in this area is the ability to handle the requirement. In other words, at the output end of the system flexibility is more important than speed. At the input end of the system the picture is more complicated. In areas where high throughput is to be expected, speed is more important than flexibility. For example, the basic screens used for processing recruitment administration, manipulating the salary review, or displaying information on individuals which would previously have been kept on the record cards, must operate fast. Under the manual svstem, if someone phones up and asks, for example, for the date of Mr. Brown's salary change, the answer would be given immediately over the phone. The speed of response to such questions which is possible under a manual system must be at least matched and if possible improved upon by a computerised system. It is no good saying that you will have to phone the person back. It will also be no good if it takes longer to change the basic details of employees under the computer system than it did under the manual system, because this will simply frustrate the people who are trying to use it. The system will fall into disuse and disrepute. As far as recruitment is concerned, of course, speed is fundamental to achieve the objectives of the system.

In the Foster Wheeler system the average time required to update an applicant (i.e. update the data base and generate a letter) is 7½ seconds. The longest operation by a long way is putting new applicants on to the system, because this requires the entering of name, address, vacancy, etc. One person can put on 100 new applicants in an hour. Where speed is as important as this, flexibility must be compromised to some extent. It is not feasible, for example, to write a system which includes the complexities discussed above (integrated letter writing and the problems of applicants who pursue peculiar routes) and to provide the speed required by using an application generator. In view of the complexities involved, the speeds involved can only be achieved by writing the system in a traditional programming language such as Fortran. Although this does not mean that the system is completely inflexible, it will necessarily mean that the input and processing side of the system is significantly less flexible, certainly as far as users are concerned, than the output side, which depends on a report generator.

Nevertheless it should be noted that flexibility of screen design is not likely to be a key requirement in a personnel system. How often do Personnel Departments redesign their manual record cards? Most of them have never done so, so why should they want to redesign their screens?

There may be areas within the personnel records system where flexibility at the input end is required and speed is not critical, for example, a very detailed inventory of skills of people within the company will probably only be updated once a year for each individual. Throughput is therefore low. The chances of wanting to include different skills at different stages in the development of the system are high and so flexibility is important. Enquiries are unlikely to be made on an individual basis. They are more likely to follow the form, "Who have we got with the following characteristics who has a given mix of skills?" Such a question would be answered through the report generator.

Since flexibility is important in this case and response times are not, this is a part of the system that could be written using an application generator. The important point is not to fall into the trap of attempting to write the core of the system with an application generator, as this will lead to the system being rejected by the people who are supposed to use it on the basis of inadequate response times.

To conclude, the HP3000 with Image 3000 is an excellent basis for a personnel system. Such a system should be written in a fast language and must include integrated letter writing with report generator and application generator facilities available where appropriate. The complexities of systems analysis and procedural analysis are considerable, and argue in favour of buying a package.

Note 1: Further information on the detailed requirements of personnel systems may be found in "Personnel Computer Systems" by Tony Ive, published by McGraw Hill, 1982, obtainable from McGraw Hill Book Company (UK) Ltd., Shoppenhangers Road, Maidenhead, Berks, SL6 2BU. Price £11.95 plus £1.20 postage and packing.

Note 2: Further details about the Foster Wheeler computerised personnel systems, which are available for purchase, may be obtained from Tony Ive, Manager Personnel Projects, Foster Wheeler Energy Ltd., Foster Wheeler House, Station Road, Reading, Berks, RG1 1LX, Telephone (0734) 585211.

A paper to be presented at:

HP3000 International Users Group,
International Conference,
Scotland.

2nd - 7th October, 1983

COMPANIES STATUTORY BOOKS

ON THE HP3000

By: Mr. N. Bedford
L.C.P. Group Services Limited

CONTENTS

ITEM		PAGE
1	.. .. INTRODUCTION	2
2	.. .. L.C.P. - WHAT IS IT?	3
3	.. .. .. STATUTORY BOOKS WHAT ARE THEY?	4
4	.. .. .. REASON FOR SYSTEM AND REQUIREMENT	6
5	USE, PROBLEMS AND SOLUTIONS USING IMAGE AND KSAM	10
6	.. .. KEEPING THE AUDIT TRAIL	19
7	.. .. SECURITY	21
8	.. .. BENEFITS GAINED BY THE SECRETARIAL STAFF	22

1.0. INTRODUCTION

Keeping the Statutory Books up to date and producing the forms required by the Registrar of Companies for a limited company with over 100 UK based subsidiaries can be a lengthy and arduous task. Having failed to find a package to satisfy the companies requirements, it was decided to build an in-house application around a complex Image Database. This paper will try to describe some of the problems and solutions encountered and overcome during the design and writing of this application.

2.0. L.C.P. - WHAT IS IT?

L.C.P. is a large group of diverse companies mainly based in the UK, but also in mainland Europe and U.S.A. The major activities of the group are outlined below:-

L.C.P. HOLDINGS plc

PROPERTY DEVELOPMENT & CONSTRUCTION
Trading Estate Management
Offices

VEHICLE DISTRIBUTION
Main Dealers for FORD, GENERAL MOTORS, BRITISH LEYLAND, ROLLS ROYCE

DISTRIBUTION
Building Supplies
Solid Fuel

Automative Parts Retailing

Most of the above companies data processing requirements are serviced by an in-house bureau based on a HP3000, with modems and multiplexors to regional depots.

Last year the group had a turnover of £277 million.

3.0. STATUTORY BOOKS WHAT ARE THEY?

It is the responsibility of the Secretarial Department based at corperate head office to maintain the statutory records for every subsidiary company and to ensure that the requirements of the Companies Acts are complied with as to the information recorded, the form in which this information is kept, the rights of inspection etc.

The statutory records to be kept by all companies are:-

1. The Register of Members

   Contains details of the companies shareholders, their current shareholding and also details of any Transfers/Allotments.

2. The Register of Charges

   Is a record of mortgages or other charges entered into by the company to secure debts.

3. The Register of Directors & Secretaries

   Contains details of directors and secretaries for the company. The details kept are dates of appointment/termination, name and previous name, date of birth, address, nationality, business occupation and any other directorships held in a non group company. Details of new appointments or terminations have to be sent to companies house on the Form 9b.

4. The Minute Book

   Is a record of proceeding at board meetings.

5. The Register of Directors Interests

Is a record of the number of shares which are held by Directors of the company. (Within the L.C.P. Group there are none).

6. The Accounting Records

Obviously the financial accounting department maintains the accounting records of the company but they fall within the formal company procedures to enable the Annual Report and Accounts to be filed with the Annual Returns.

Each company must complete and file an Annual Return (Form 6A) which gives the information contained in the registers No. 1 - 3 above, together with a summary of it's Nominal & Issued Share Capital, the Registered number and situation of the Registered Office.

With the exception of accounting records the statutory registers have traditionally been maintained by handwritten entry in bound books, but the development of companies has lead to the acceptance by the Registrar of companies to the holding of these records on computer. Recent legislation 'The Companies (Registers and other records) Regulations 1979', lays down the procedures which must be followed in respect of registers kept other than in legible form.

Continued...

4.0. REASON FOR SYSTEM AND REQUIREMENT

a) Why was it decided to computerise the Manual System?

As mentioned previously the companies statutory books were kept in hand written bound books of which there are over a hundred. A change to any of the held details would require a hand written amendment. Any errors made would have to be crossed out possibly resulting in many books from which facts and figures could not quickly and easily be retrieved.

When producing the yearly Annual Returns Form 6A many companies retained the same details as those of the previous year, even so a new form had to be prepared and typed. This meant that in preparing the Annual Returns details, time and effort were duplicated.

The tasks of keeping details up to date and producing the forms required by the Registrars Office were, for the staff involved, laborious and time consuming. Therefore details were more readily prone to error. For the reasons already mentioned and having failed to find a suitable package, it was decided to build an in house application around a complex Image Database.

Continued...

Continued...

b) Which parts of the Manual System were Automated?

The main area in which automation would achieve savings was the production of the formal documents that were required by the Registrar of Companies. These documents are:-

Form 4a, Notification of a change in address of the registered office.

Form 9b, Notification of an appointment or termination of a director or Secretary.

Form 6A, The Annual Return of a company having a share capital.

To produce these documents certain registers kept by the manual system would have to be mirrored in the database.

The registers were:-

a) The Register of Members
b) The Register of Charges
c) The Register of Directors and Secretaries
d) The Register of Transfers

In addition to these, details would also have to be kept about the share capital for each company as this is not shown in any of the above registers.

Conti ..

It can be seen that not all of the manual system was automated, the reasons for this are as follows:-

The Director Share Holding Register is not kept by L.C.P. for their subsidiaries, as at present, no director holds shares in these companies. This register, the accounts of companies and minutes of the Annual General Meetings are not required to complete the Annual Returns, although the accounts and minutes are required by the 'Registrar of Companies'. It is proposed that at a later date these parts of the manual system will eventually be automated.

c) What would the System produce?

Internal Registers

In addition to the documents produced to send to the Registrar of Companies there is a legal requirement that the registers can be inspected by shareholders. This used to be a simple matter of obtaining the right book and opening it at the required page. It was, therefore, necessary to provide a facility to produce reports of the kept registers at the Secretarial Department's request.

These reports were for internal use and were, therefore, only subject to the secretarial department's approval.

External Formal Documents

The Registrars office has only recently agreed to accept the Forms 4a, 9b, and 6a on computer stationery. Previously all details were typed on their pre-printed forms. A set of guidelines for the Form 6A were obtained, but we were also informed that these might change in the near future.

Enquiry Facilities

With details for every subsidiary being held on one database, enquiry programs would replace the time and effort needed to obtain details concerning information held in the Statutory Books. For example, an enquiry as to which subsidiary a person was a director of could be produced by entering just the surname. In the manual system the secretarial staff would have to look in every Statutory book held.

To reduce the D. P. Staff's involvement in the running of the application all programs to produce reports, enquiries, forms and maintenance of details would be MENU driven. A menu was produced to deal with each part of the system, i.e. one menu relating to Directors details, one for the allotment and transfer of shares, etc. The only D. P. Staff involvement would be in the distribution of reports and forms and the maintenance of the system.

5. USE, PROBLEMS AND SOLUTIONS USING IMAGE AND KSAM

a) Why use an Image Database?

The details that are needed to produce the required results from the system are closely related. All of the registers to be automated were needed to produce the Annual Returns Form 6A. Other details from these registers were needed to produce the forms 4a and 9b. It therefore seemed logical to hold all the required details centrally on an Image Database.

The choice of using an Image database meant that separate files need not be kept for each register. The design of the database is such that no details for any one register are kept on a single data set. For example by using a Master data set for a Director's name, address, nationality, and date of birth there will only ever be one entry for these details. Due to the fact that a director can hold an appointment in many companies, a detail data set can then be used with company number as the search item to hold appointment and resignation dates.

Common details held in this manner can be updated very much easier and only need to be done once. This of course reduces the number of accesses needed to carry out an action, compared to those that would be needed had these details been held in a non-centralised way. By holding common data once, also reduces the proliferation of entries due to the reduction of redundant details.

The decision to hold details centrally aided the number and type of enquiry programs that could be produced for the secretarial staff. In other systems currently being used it was necessary to run unload programs overnight to allow the use of certain enquiry programs next day. This was not required in the case of statutory books because of the above reasons, also any KSAM files required for generic searches are maintained automatically when changes to the database are made.

b) Why Use KSAM files?

Approximately two years ago enquiry programs started to be written for the user department. It was found that a generic search could not be carried out on a database using the available intrinsics. This meant that to use a generic search on a required key, the details of such a key had to be unloaded to a KSAM file overnight. To reduce this overhead on the system it was decided that any keys required by generic searches, would be automatically maintained on a KSAM file at the same time as the database.

There are only two KSAM files used by the system, one for director's surname, the other for members surname. The records are only 76 bytes long and there are not that many entries, therefore, the files are quite small.

c) Choice of Keys - Problems and Solutions of Maintenance

If the database had been designed so that each register occupied one data set there would only have had to be one major key, this being Company Number. However, the registers details are spread over a multitude of data sets, the keys and search items needed to formulate these registers are, therefore, more numerous. Below is a list of the chosen keys.

i COMPANY NUMBER
ii DIRECTOR/SECRETARIES SURNAME
iii MEMBERS SURNAME
iv COMPANY NUMBER + MEMBER NUMBER
v AUDIT RECORD KEY

i COMPANY NUMBER

An Annual Return, Forms 4a, 9b and the registers are all produced for particular companies, therefore, there must be a way of distinguishing between them. A number was allocated to each company by the Secretarial Staff before adding details to the database. The company numbers are four characters in length, thereby allowing up to 9999 numbers to be allocated. It was decided to split this number into ranges, each range allocated to a division of the group, for example 2001 to 3000 Construction Division. A range of numbers were also allocated to non subsidiary companies with which persons held the position of director.

Continued...

If an error was made when entering the company details, ie. the wrong company number was given to a company it can be changed at this point. After the details for the registers have been added it was not possible to amend the company number without possibly corrupting the Members Share and Transfer Registers.

If a company's details had to be deleted, i.e. it was sold, it is imperative that the details of the registers are deleted first. The reason for this being a chain head cannot be deleted before the associated details on the detailed data set have been removed. This does not present any real problems to the Secretarial department, for a deletion of this kind is very rarely carried out.

Continued...

Continued...

ii DIRECTOR/SECRETARIES SURNAME

By using the surname of a director/secretary it would be familier with the secretarial staff, a number or other identifyer would have had to be assigned and kept up to date. One of the reasons for the system was to reduce the amount of work involved not increase it. In the first place only the surname was used, but it was noticed that there happened to be two directors with the same surname. To overcome this, it was decided to add the first character of each christian name to the key, this then made it unique.

The key can be amended by the secretarial staff quite easily, as for the program, thats a different matter.

The surname is the key to a Master data set and also a search item to a Detail data set. This meant that changing a key entry is not just a matter of deleting the old entry and creating a new one. The procedure taken by the program to complete this task is shown below:-

```
(OLD KEY)    READ OLD MASTER ENTRY
(NEW KEY)    WRITE NEW MASTER ENTRY
(OLD KEY)    SET UP CHAIN          )
(OLD KEY)    READ OLD ENTRY        )
(OLD KEY     DELETE OLD ENTRY      )   CONTINUE UNTIL
(NEW KEY)    WRITE NEW ENTRY       )   ALL OLD ENTRIES
(OLD KEY)    READ OLD MASTER ENTRY     EXHAUSTED
(OLD KEY)    DELETE OLD MASTER ENTRY
```

Continued...

Continued...

The chain has to be set up after every read, delete and write to the detail data set. The reason for this is that Audit Records are written after the "write" of the new entry. When writing the Audit Records other data sets have to be locked and unlocked which destroys the chain pointers, for as yet there is no way to unlock a specific data set. One unlock will release all outstanding locks, which include the one covering the chain pointers.

The above procedure was not only done for a change in Surname, but also had to be done for any change in the first two christian names.

The deletion of a Director/Secretary from the database required a similar procedure to the one above, again the unlock was the problem.

In addition to changing the database, a KSAM file also has to be updated at the same time. This file contains the Director/Secretary surname plus the first line of the address and is used for generic searches by the enquiry programs.

iii MEMBERS SURNAME

The Members Surname was chosen as a key for the same reasons as the Directors Surname. Because the Members Surname would be used more often it was decided to allocate each member a unique number. The surname was, therefore, only an interface between the system and secretarial department. The number allocated to the surname was called the member number and is used in most data sets, this also allowed a much smaller combination of fields to make up a sort item on one of the detailed data sets.

The problem with the directors surname not being unique never arose with the member surname. There are only two members of each subsidiary company, one is 'L.C.P. Holdings Plc' and the other a nominee. The nominee surname is added to the subsidiary company name thereby making a unique key. Each subsidiary company only has two share holders as required by Law, there are no future proposals to increase this number.

The creation, amendment and deletion of this key is carried out in a similar manner to that of the directors programs mentioned previously. A KSAM file is also kept containing the surname and first line of address, this is then used for generic searches by the enquiry programs.

Continued...

iv COMPANY NUMBER AND MEMBER NUMBER

This is a composit key used by only one data set. All details relating to allotment and transfer of shares are held on a detail data set in a 'sorted' order. Part of the sort item is the date of allotment or transfer, therefore, it is consievable that if a running total is kept of total share holding and an amendment made to this date, the total share holding figure will no longer be correct. This will also hold true for the deletion of any entry.

It was, therefore, decided to hold the total shares of a member, for each subsidiary company on a separate master data set. The key of this data set would be made up of the Company and Member Numbers associated with the required total. This total could also be used by the enquiry programs when details of members share holdings were required by the secretarial department.

The creation of an entry is done automatically when a member is allocated shares, and removed when or if a member is deleted. The key is amended quite simply by deleting the old entry and creating the new one with the required key. This action is taken only after checking that the new required key is not already in existance on the database.

Continued...

v AUDIT RECORD KEY

An Audit Record is written to the database for every action (Creation, Amendment, Deletion) carried out on a record or field. These details are then used at a prescribed time to produce an 'Audit Trail Report'.

When an Audit Record is written the record number on the Audit Control Data Set is incremented by one. This number is then used as the key to an Audit Record that is written to the database.

When the 'Audit Trail Report' is produced the record number on the Audit Control Data Set is set to zero and all Audit Records are deleted from the database using Query.

Continued...

6.0. KEEPING THE AUDIT TRAIL

As previously mentioned each register was not kept on a single data set but consisted of one or more master and detail data sets. This may have reduced the amount of redundant details and number of entries but it also made the database more complex. The programs written to maintain the details were, therefore, more complex in design. Large and complex programs bring into play other questions such as maintenance, this topic will not be covered in this paper.

The keeping of an Audit Trail for any changes in data on the database is a departmental standard. This means for every record created, amended or deleted an Audit Record is written to a master data set.

The method used to write an audit record is shown below:-

```
LOCK AUDIT CONTROL
READ AUDIT CONTROL
    ADD 1 TO RECORD NUMBER
UPDATE AUDIT CONTROL
UNLOCK AUDIT CONTROL
LOCK AUDIT SET
WRITE AUDIT RECORD
UNLOCK AUDIT SET
```

Continued

Continued...

This method of keeping the Audit Trail causes no problems when actions are carried out on Master Data Sets. The problems encountered with locking and unlocking data sets, were in changing the key on a detail data set, this has been mentioned previously along with one solution to the problem. Another solution used to overcome this problem was to give the program M. R. (Multi-Rin) capability. This allowed the programmer to keep a lock on the detail, Audit Control and Audit Data Sets while amendments were being carried out.

When the end of chain condition was met on the detail data set one unlock is carried out, clearing all data set locks in one operation. The main problem with this method is that no other process using the database can write an Audit Record. This was only a small price to pay, as this is not a high volume transaction system.

Continued...

7.0. SECURITY

The security of the system is achieved by a series of passwords. An account password is required from the secretarial staff to access the main menu of the system. Each program in the system chosen from a menu then requires another password before entry to the program is gained.

In order to protect the more sensative details such as the amendment of allotments and transfers a second program password has to be entered, failure to do so will return the user back to the menu.

All actions carried out to records or fields are written out to the audit data set. The password used to access a program to carry out such actions identifies the user. This identifier in terms of an operator name is included on the audit record. By looking at the 'Audit Trail Report' it can be seen who carried out what actions, to which details, at what date and time.

The enquiry programs used in the system also carry an extra security. If after a period of three minutes a member of the secretarial department has not made a response to the program prompts, they are returned to the menu. This is done using timed reads within the program.

8.0. BENEFITS GAINED BY THE SECRETARIAL STAFF

The first benefit gained by automating the Statutory Books Manual System was the production of the Annual Returns Form 6A. Any changes to the details held on the database are made by the secretarial staff, the form 6A will then automatically pick up these changes and include them in the print. In the previous manual system any changes had to be made twice, firstly in the actual statutory book and secondly on the Annual Returns Form 6A.

The time and effort needed to produce the Form 6A has also been reduced. In the manual system, the previous years form 6As are updated with any relevant changes then passed to a secretary for typing. On completion they are stringently checked for any errors, when correct they are forwarded for signing. A photo-copy is then taken and the originals sent to the Registrar of Companies. With the automated system an edit is first produced, once checked, the Form 6As are produced on two part paper and forwarded for signing, the top copy is then sent to the Registrar of Companies. There is a considerable saving in time and effort in producing the Annual Returns now that they have been automated.

Continued...

Continued...

Benefits have also been found in the production of the Forms 4a and 9b. Once changes relating to these forms have been made on the database, the forms are automatically produced on two part stationery, signed, and the top copy sent to the Registrar of Companies. With the manual system the procedure was similar to that for the Annual Returns, mentioned previously, with the exception of using the previous years forms.

A problem that was noted when using hand written bound books to hold statutory information, was the ease with which these books could be made ilegible, and for the older companies become full. The automated system does not have the problem of untidyness or the registers of the older companies overflowing.

The registers of a Company can be produced in a clear and correct format at the press of a button (or two).

One of the problems with keeping the Statutory details for a large number of subsidiary companies was that if a common entry i.e. a Director's address, was changed, all statutory books containing this address would have to be amended by hand. Because these type of details are only held once on the database, only one change is required in the automated system. This again reduced the time and effort involved in making such a change.

Continued...

In automating the system there is now the facility to produce information quickly using enquiry programs. This is very useful in answering questions, such as "For which companies is Mr. Bloggs a Director?" To answer this type of query using the manual system, would have resulted in every Statutory Book kept being examined for this entry.

Some of the details held on the database are not required for statutory reasons, but is of use to the secretarial staff. These details are shown below:-

COMPANY TRADING NAMES
COMPANY TRADING ADDRESSES
DIVISIONAL RESPONSIBILITIES e.g. FINANCIAL DIRECTOR
CHAIRMAN etc.

These details can be used to produce reports when required by the secretarial staff for internal use.

The benefits that have been gained by the secretarial department, in automating their manual system, seem to have been worth the time taken, and effort involved.

Grampian Software Facilities is the software division of Grampian Computer Facilities Ltd. Established in 1970 Grampian rapidly became a respected and efficient Computer Bureau in Aberdeen in the North East of Scotland.

Grampian has been involved in the design of many systems for local and national companies and in 1979 became a Hewlett Packard O.E.M. Since that time over 30 250's and 3000's have been installed and also software supplied to third party Hewlett Packard users.

The company has specialist software for the Whisky Industry, Construction Industry, Payroll and General Accounting and has developed a Purchase Ordering and Stock Control system for the North Sea Oil Industry. All of this software will be available for demonstration during the IUG meeting but the application to be discussed in this presentation is payroll.

H-PAY

H-PAY is an interactive on-line payroll system developed for the Hewlett Packard 3000 range of computers. The system was originally released in 1980 and Mark 2 was released in early 1983.

Agreement has been reached between Hewlett Packard and Grampian on the joint development of an enhanced version of H-PAY for use on HP's 3000 systems. No firm release date has been fixed but both companies are working toward the package being available near the end of the year. This version is currently being referred to as H-PAY3. Progress to date is good and during 1984 the product will be marketed by both HP and Grampian.

The system described in this presentation is H-PAY2, but where relevant, H-PAY3 enhancements will be highlighted.

There are currently 18 users of H-PAY and approximately 20000 individuals depend on H-PAY to provide them with a payslip. The list of Users is shown later but it is worth noting that over 900 HP employees of the Telecommunications division in South Queensferry are paid using H-PAY.

What makes H-PAY different from other payroll systems? - We think that it is better because of the following reasons:-

1. The system was designed by people with a vast knowledge of payroll requirements in a bureau environment gathered since the mid 1960s.

2. It was written specifically for the 3000 using IMAGE and other HP utilities. It is not a batch system with an on-line front end.

3. The use of VIEW screens allows the User or Grampian to tailor input to match source documents easily and to add extra validation checks.

4. All update programs are re-runnable, thus avoiding the necessity of extra security back-ups.

5. Processing is fast and efficient - a necessity for the overworked payroll department.

6. 100 elements of pay are available for each payroll.

7. Elements of pay can be accumulated in 35 different accumulators in 4 different time periods to cover Pension Year, Employer Year, Tax Year and Holiday Year.

8. Payroll costing is an integral part of the system.

9. Interfaces are provided to other systems, such as Contract Costing, Nominal Ledger, etc.

10. A simple Report writer provides Payroll, Personnel and Management with a wide range of reports.

11. A Statutory Sick Pay and absence recording system is incorporated.

H-PAY3 will have the following additions:-

1. A multi-payroll database allowing all types of payroll within a company in the one database.
2. Allied to this there will be a new MENU/SUPERVISOR system with extensive security and passwords.
3. Additional features to allow for calculations based on averages, previous earnings etc.
4. A wider range of Personnel fields. (H-PAY3 will eventually interface with a complete Personnel system).
5. The number of pay elements will be increased to 200, with 100 more reserved for future use.
6. The number of accumulators will be increased to 80.
7. An improved Report Writer enabling Management reports to be produced more easily.

System Concept

The system uses an IMAGE database - and a KSAM file to speed up the production of reports, enquiries etc.

Each payroll is currently a separate database (to be changed in H-PAY3) and the initial creation of a payroll is carried out by the Manager of the Payroll account by creating a database using the Schema and Jobs supplied with the system.

The payroll details such as elements of pay, accumulators used etc can either be set up from scratch or copied from another payroll.

After the company details have been finalised and any elements/accumulators which are unique to that payroll have been completed, cost centres are set up which are valid for the employees in the payroll and again these can be copied from an existing payroll.

Employee records are created either by direct on-line input or by conversion from an existing system. Records are validated as they are input and comprehensive error messages keep the User right. As an employee record is created, an entry is made in the KSAM file.

At the end of the current session the employee records which have been created or amended are either printed or written to a cumulative print file for subsequent printing.

Employee timesheet or clockcard input can be added to the database on-line using HP block mode terminals or can be loaded from an MPE file created by either an integral or stand alone Data Entry system. Input can be made for either the CURRENT or NEXT pay period.

The next stage in the input procedure is to update the SSP and absence records if any. Full details of the SSP module follow later.

The User is now almost ready to process the payroll, but there may be employees who are not to be paid, employees to be taxed at a different tax period etc and for these employees a Variance will be required which is entered at the same time as the payroll run header.

The payroll is now ready to be processed. The calculation program can either be processed as a session or as a stream job and calculates gross to net for all employees at a speed of approximately 60-100 employees per minute, depending on the size of the machine and its loading. This process can be repeated as often as necessary during the current period so that restoring from the previous week's files is not necessary in the event of a re-run.

The User now has the option to produce either the normal payroll reports, payslips, a BACS tape or other special reports. The order in which these are processed will depend on the requirements of the payroll section and the User has complete control over this.

At the end of the processing period, or possibly just before the start of the next one, it is necessary to process an "End of Period Run". This updates the database cumulatives, zeroises the payslip and current week timesheet details, writes out history files and prepares the masterfile for the following week or month's input.

Statutory Sick Pay

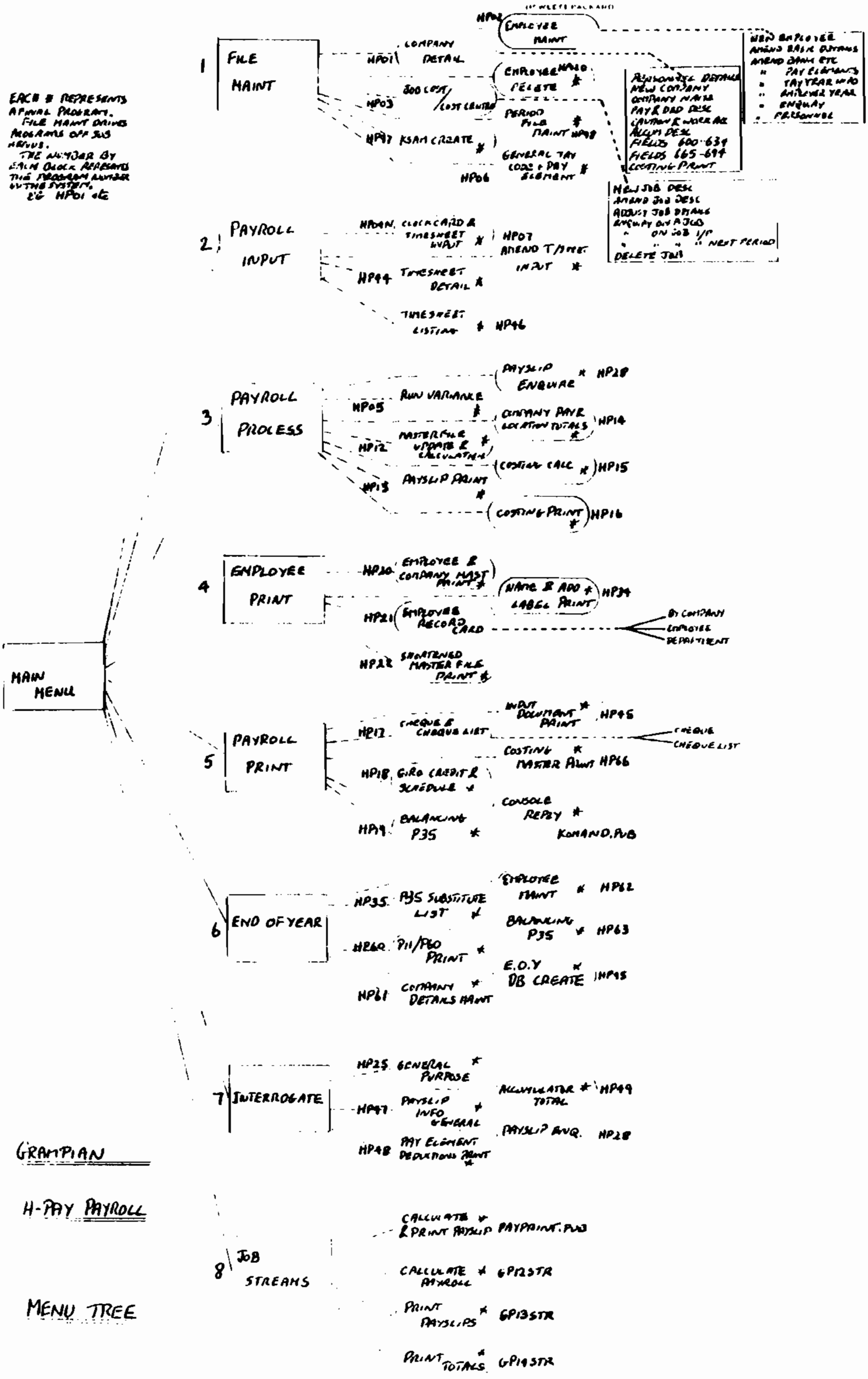
The Statutory Sick Pay module in H-PAY allows the User to monitor and control all employee absence, whether liable to SSP or not. Gross Pay is maintained over a 13 period cycle and average earnings calculated over the last eight of these so that the appropriate SSP rate can be paid. If necessary the average can be calculated on any specified periods. For monthly payrolls the average is calculated on the last two months.

Qualifying days are held at payroll level and can be amended at employee level and therefore the daily rate for SSP can be displayed on the SSP input screen. The User inputs the periods of absence and the system calculates the SSP due prompting the User if the various limits are exceeded. The amount of SSP is either paid to the employee or offset against an existing payment. If the number of days of absence input does not qualify for SSP the User can indicate this and maintain a full absence record. Various print-outs provide Payroll and Personnel departments with full audit records for DHSS purposes.

LIST OF H-PAY USERS

Company/Contact	Location/Telephone	Machine
Kestrel Marine Ltd W Craigie	Dundee 0382-457575	3000
John Fyfe Ltd D Norrie	Aberdeen 0224-691333	3000
A Hall & Son (Builders) Ltd J Paterson	Aberdeen 0224-693155	3000
North Eastern Farmers Ltd J McGregor	Aberdeen 0224-21255	3000
Rigblast Holdings Ltd B Chisholm	Aberdeen 0224-722888	3000
A M International Ltd R Bevins	Hemel Hempstead 0224-42251	3000
Caribonum Ltd R Kinrade	Peterborough 0733-234737	3000
Admel Ltd D Morris	Weybridge 0932-47212	3000
The Invergordon Distillers K Kinnaird	Glasgow 041-248-3051	250
J Fraser Construction Ltd R Crawford	Dundee 0382-737721	250
Hewlett Packard Ltd J O'Neil	South Queensferry 031-331-1000	3000
Drake & Scull Engineering Ltd S Henry	Salisbury 0722-24611	3000
William Tawse Ltd G Cumming	Aberdeen 0224-742222	3000
Houlder Offshore Ltd M Forbes	Aberdeen 0224-714101	3000
Hunter Construction Ltd H Matthews	Aberdeen 0224-873363	250
Santa Fe (UK) Ltd S Riddoch	Aberdeen 0224-871747	3000
Golden Ltd (L'Oreal) B Clee	South Wales 0443-223456	3000

Company/Contact	Location/Telephone	Machine
Span Construction Ltd R Birch	Aberdeen 0224-820465	250
Barton Distilling	Alexandria 0389-52555	3000


MAIN MENU
1 FILE MAINT
HEWLETT PACKARD
HP02 EMPLOYEE MAINT
NEW EMPLOYEE
AMEND BASIC DETAILS
AMEND BANK ETC
" PAY ELEMENTS
" TAX YEAR INFO
" ANNUAL YEAR
" ENQUIRY
" PERSONNEL
HP01 COMPANY DETAIL
EMPLOYEE HARD DELETE *
HP03 JOB COST/COST CENTRE
PERIOD FILE MAINT HP08 *
HP97 KSAM CREATE *
GENERAL TAX CODE & PAY ELEMENT *
HP06
PERSONNEL DETAILS
NEW COMPANY
COMPANY NAME
PAY & DED DESC
CAUTION & WORK AC
ACCUM DESC
FIELDS 600-634
FIELDS 665-694
COSTING PRINT
NEW JOB DESC
AMEND JOB DESC
ADJUST JOB DETAILS
ENQUIRY ON A JOB
" ON JOB I/P
" " " " NEXT PERIOD
DELETE JOB
2 PAYROLL INPUT
HP04N CLOCKCARD & TIMESHEET INPUT *
HP07 AMEND T/SHEET INPUT *
HP44 TIMESHEET DETAIL *
TIMESHEET LISTING * HP46
3 PAYROLL PROCESS
PAYSLIP ENQUIRE * HP28
HP05 RUN VARIANCE *
COMPANY PAYE LOCATION TOTALS * HP14
HP12 MASTERFILE UPDATE & CALCULATION *
COSTING CALC * HP15
HP13 PAYSLIP PRINT *
COSTING PRINT * HP16
4 EMPLOYEE PRINT
HP20 EMPLOYEE & COMPANY MAST PRINT *
NAME & ADD LABEL PRINT * HP34
HP21 EMPLOYEE RECORD CARD
BY COMPANY
EMPLOYEE
DEPARTMENT
HP22 SHORTENED MASTER FILE PRINT *
5 PAYROLL PRINT
INPUT DOCUMENT PRINT * HP45
HP17 CHEQUE & CHEQUE LIST
CHEQUE
CHEQUE LIST
COSTING MASTER PRINT * HP66
HP18 GIRO CREDIT & SCHEDULE *
CONSOLE REPLY *
HP19 BALANCING P35 *
KOMAND.PUB
6 END OF YEAR
EMPLOYEE MAINT * HP62
HP35 P35 SUBSTITUTE LIST *
BALANCING P35 * HP63
HP60 P11/P60 PRINT *
E.O.Y DB CREATE * HP95
HP61 COMPANY DETAILS MAINT *
7 INTERROGATE
HP25 GENERAL PURPOSE *
ACCUMULATOR TOTAL * HP49
HP47 PAYSLIP INFO GENERAL *
PAYSLIP ENQ. HP28
HP48 PAY ELEMENT DEDUCTIONS MAINT *
8 JOB STREAMS
CALCULATE & PRINT PAYSLIP * PAYPRINT.PUB
CALCULATE PAYROLL * GP12STR
PRINT PAYSLIPS * GP13STR
PRINT TOTALS * GP14STR

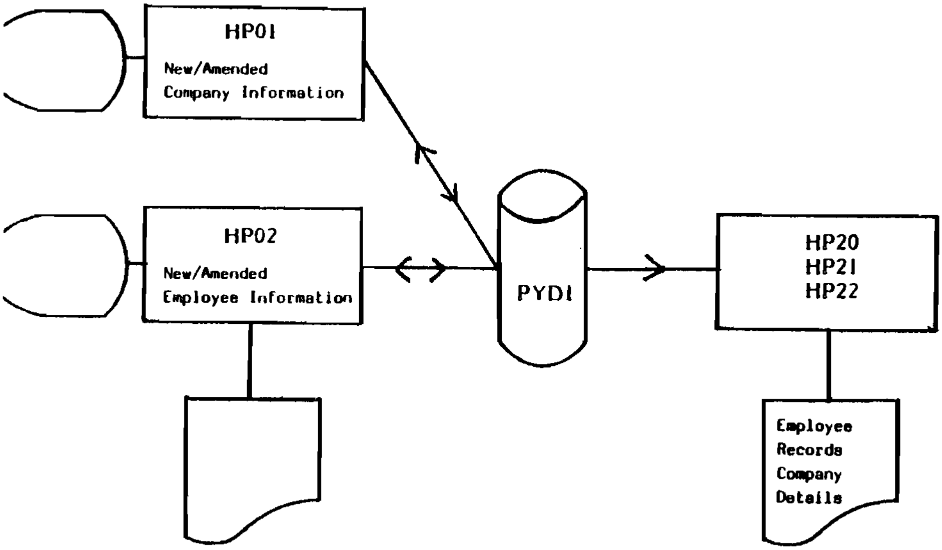
EACH * REPRESENTS A FINAL PROGRAM.
FILE MAINT DRIVES PROGRAMS OFF SUB MENUS.
THE NUMBER BY EACH BLOCK REPRESENTS THE PROGRAM NUMBER IN THE SYSTEM.
EG HP01 etc

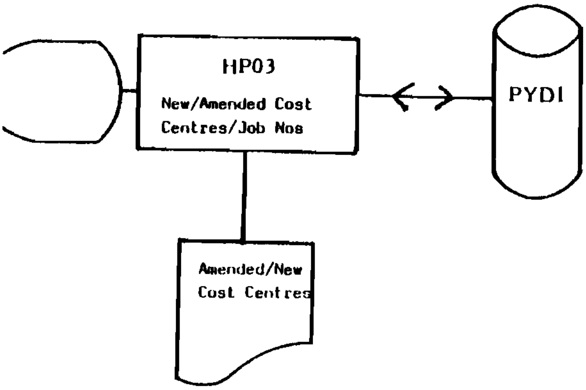
GRAMPIAN

H-PAY PAYROLL

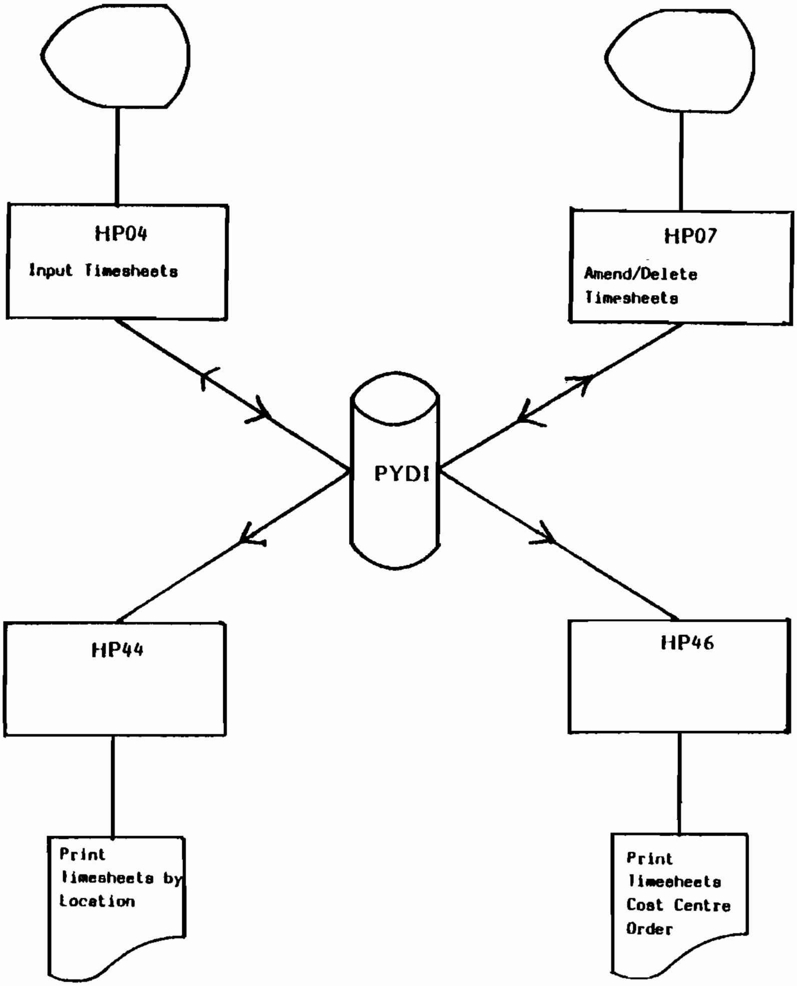
MENU TREE

H-PAY
MAINTENANCE

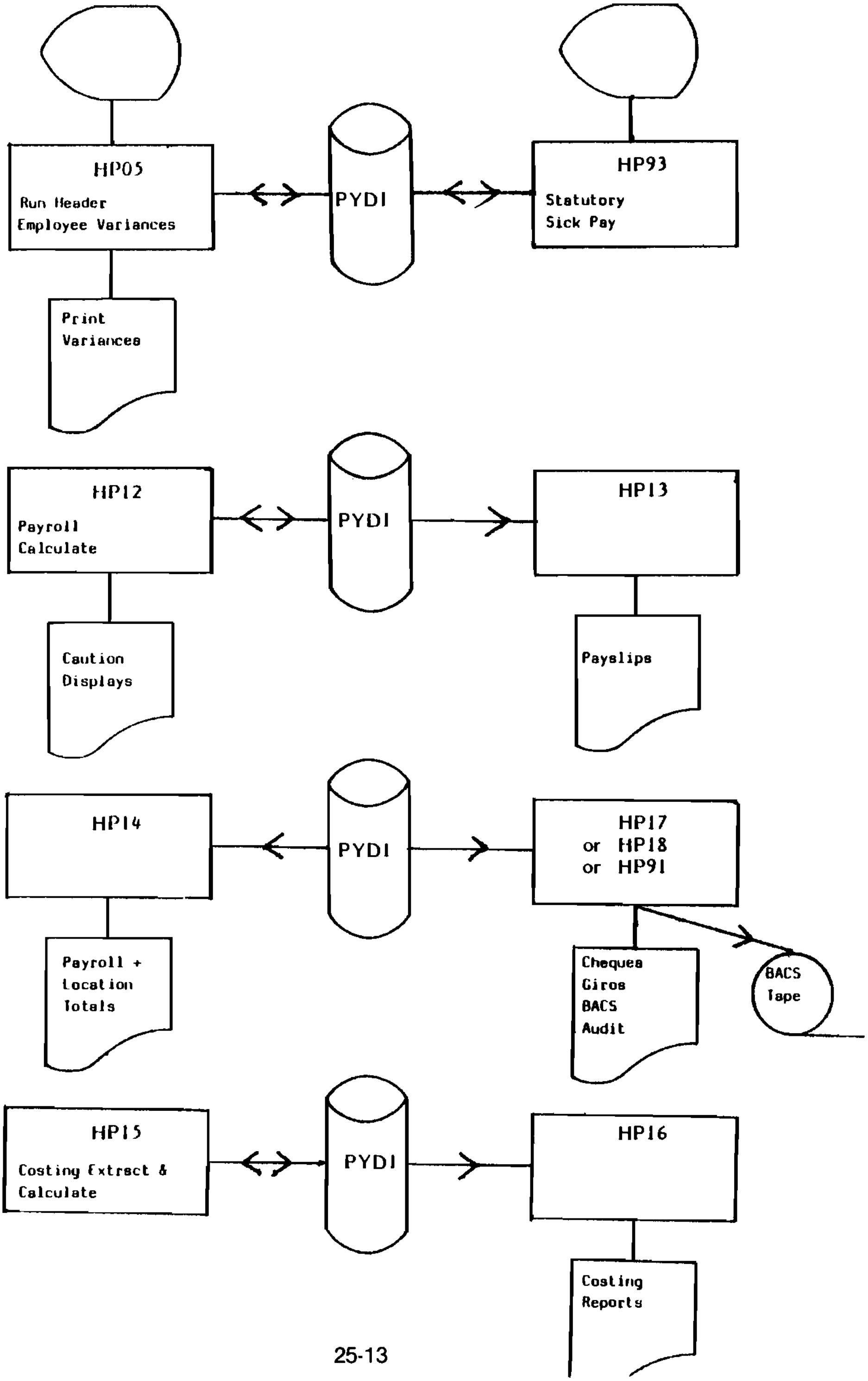

HP01
New/Amended
Company Information
HP02
New/Amended
Employee Information
PYD1
HP20
HP21
HP22
Employee
Records
Company
Details


HP03
New/Amended Cost
Centres/Job Nos
PYD1
Amended/New
Cost Centres

H-PAY
TIMESHEET INPUT


HP04
Input Timesheets
HP07
Amend/Delete Timesheets
PYDI
HP44
HP46
Print Timesheets by Location
Print Timesheets Cost Centre Order

H-PAY
PAYROLL PROCESSING


HP05
Run Header
Employee Variances
PYDI
HP93
Statutory
Sick Pay
Print
Variances
HP12
Payroll
Calculate
PYDI
HP13
Caution
Displays
Payslips
HP14
PYDI
HP17
or HP18
or HP91
Payroll +
Location
Totals
Cheques
Giros
BACS
Audit
BACS
Tape
HP15
Costing Extract &
Calculate
PYDI
HP16
Costing
Reports

Version 1/0783

"PERSON - the personnel system that's different"

1. INTRODUCTION

Although a Personnel Information System is in fact a specific Applications Package, the Conference will probably find it useful to hear of some of the techniques that have had to be employed in order to support the generalities that such a system requires.

In conducting the initial market and feasibility survey it soon became apparant that the variety of information that needed to be stored against individuals was enormous. Each Company surveyed had a requirement to access standard data (names, addresses, skills, job history) along with its own selection of industry-specific data. Examples are: detailed educational background for Colleges; tool and machine relevant data for Engineering Industry; allergies for Medical Data - the diversity was endless.

One obvious consequence to this was that different Personnel Officers in each of the different diciplines wanted their own specific selection of data on their own screen. They also wanted reports which included their own relevant combination of data from hundreds of different data items.

This is usually a job for a fourth generation language, but, as far as a Personnel Officer is concerned, you can't get much more User Unfriendly than to start discussing Pictures, Data Set Relationships, Compilations and the like. He doesn't want to become a "Programmer" and, no matter how high level the language is, this is what is being asked of him.

The problem became, therefore, one of providing an Application Package sufficiently flexible to create variable input and output screens and variable reports but without resorting to compilations/generations of programs, data bases or screens and without introducing too many unfamiliar concepts and buzz words.

In this presentation we attempt to cover some of the techniques used to introduce the necessary "Customisation" - AS THEY APPEAR TO THE PERSONNEL OFFICER. At the end of the session, we will deal with the technical side of the system, mainly by means of questions from the floor - since it will become apparant that most of the techniques are easily achievable using standard Image Data Base, Cobol and View.

We feel it is important to include the System Overview (below) and to note that this has been presented to, and understood by many Personnel Managers and Officers. It sets the scene of the level of computer jargon that is allowable. Since "data" is the stock in trade of the Personnel Officer there was absolutely no problem in the concept of Data Base in terms of an information pool and the associated terms File, Field, Data Dictionary were readily understood.

SYSTEM OVERVIEW

PERSON aims to provide a Data Base Structure to enable the storage of, and easy access to, the various types of information held against employees by Companies of widely different disciplines.

Because the flexibility required is at data item level, a system is needed that allows the user to define his OWN data items as well as to access standard data items. This is accomplished by creating a Data Dictionary of standard fields within the Data Base and then enabling User additions to this Dictionary for non standard items specific to that User.

On a similar basis standard screens are available for maintenance and enquiry purposes but it is also possible to create User defined screens containing just that information that is relevant to the User.

This User definition also applies to output reports. A later section of this Report explains in some detail how to use the User Definition facility.

The Data Base itself consists of a master file of "single fields" (the Personal set) together with several detail sets of "repeat fields" such as Associated Addresses, Education/Qualifications, Job History Information, Absence History Information. Screens of maintenance and enquiry allow access to all of these different sets of information. A further maintenance screen enables the transfer and enhancement of update information from PAYPLAN (The Standard Payroll System) to the PERSON Data Base. The use of these screens together with User defined screens is explained later in this Report.

Reporting from the system is accomplished by use of several standard reports coupled with an almost unlimited number of special User Defined Reports.

Facility has been included to enable salary elements to be changed in "batch" mode by entry of the parameters of the salary change. This facility is included in a General Update program which also accomplishes other updating of a non interactive nature.

A knowledge of the Security and Screen Conventions used is important, to understand the full flexibilities of the system. These are covered in the next section of this Report.

3. SYSTEM CONVENTIONS

3.1 SCREEN CONVENTIONS

The system is completely menu-driven using the function keys, which are also used as necessary to facilitate movement/updating WITHIN the screens.

Special Function Keys

Function Key 8 is always used to EXIT from screen in use back to the menu which routed to this screen, or exit from the Help Facility below.

Function Key 7 is the HELP key. It is used in conjunction with NEXT facility on line 23 of each screen:-

If HELP is depressed with blank in NEXT it results in the screen in use being stored and then replaced by a set of Operating Instructions applicable to that screen. When EXIT is depressed, the original screen is replaced.

If HELP is depressed with a screen number (01 - 99) in NEXT, it provides a direct route to that screen, bypassing menus.

If HELP is depressed with PT in NEXT, it results in a print of that screen to the thermal printer or attached slave printer if applicable.

Menu Structure

Function keys 1 - 6 are now available for use of routing through menus or for within-screen functions.

The screens other than MAIN MENU have a number built up from 1st digit = position of the sub menu within MAIN MENU, and 2nd digit = position of the screen within sub menu.

For Example - key 2 of Main Menu, "MAINTAIN" takes control to a Sub Menu of different Maintenance Possibilities, the 4th of which is "Employment Details". Selecting this function key gives a work screen to allow Maintenance of Employment Details, with screen number = 2 4.

3.2 SECURITY

It is important to keep a strict security on the access of information within the system, in view of the nature of the details held. For this reason each operator with access to the system must be established with an associated security code and a list of all the data dictionary items to which the operator has access. This is accomplished on a special screen for the purpose, itself having a special security code, and allowing ranges of data dictionary items to be. Associated with Operator Code and Security level.

3.3 CODED NARRATIVES

Since much of the data held is narrative (reasons for leaving/starting, comments, professions, educational establishments ...) the system lends itself to coding these narratives into Tables and then referring to the code rather than the narrative upon input. For this purpose such are completely User defined, each table being referred to by a letter of the alphabet.

At any point of input to the system, ANY alpha description/narrative can be referred to by its table letter followed by two digit code rather than by the description itself.

4. SCREENS AVAILABLE

The system is multicompany and the Main Menu screen allows entry of Company Number against Operator Code and Security Code.

After the normal check that this combination is valid the choice of submenus available is:

4.1 DEFINE PARAMETERS	allowing choice from a family of screens enabling parametric information of a general and specific nature. Amongst this family of screens are the two major screens to create the Customisation - User Defined Screen and User Defined Reports.
4.2 MAINTAIN PERSON	allowing choice from a family of screens to enable Maintenance of the various sections of the information held.
4.3 INFORM	allowing the same choice as above but in read only mode. INFORM screens cannot be used to alter the details held.
4.4 PROCESS REPORTS	which present the Processes and Reports available for selection. These include the standard Processes and Reports included within the system plus any User Defined Reports.

Interesting aspects of some of the screens available are outlined below.

5. SET TABLES (from DEFINE PARAMETERS family of screens)

Up to 26 tables of narratives against two digit codes can be established using this screen (one for each letter of the alphabet).

The "empty" tables are initially labelled A-Z and to bring a new table into action, depress the relevant function key and name the table.

The screen allows entry of "table name" which also becomes the new function key label and then allows a series of entries of two digit code against description.

Example of Use:	Table B might become JOB TITLES and code 03 could be TRAINEE ACCOUNTANT. Thereafter, at any entry point throughout the System B03 could be used as a code to mean TRAINEE ACCOUNTANT.

6. PERSON DETAILS (again from DEFINE PARAMETERS)

This screen serves as a "data dictionary" in that it is used to display all of the fields of information held on the total PERSONNEL FILE.

Information is held in two major categories:

Personal Information
Employment Information

For each category, the information held is in two further classifications:

Single Details -	information appearing once, such as name, NI number, date of birth; and
Repeat Details -	repeated information, such as qualifications, jobs held, absence history.

Along with each field the data number within the data base is shown.

Many single details and repeat details are held on the file as standard information. However, it is also possible to insert further user-defined fields using this screen.

Depress the NEW DETAILS function key. The next available data dictionary number will be displayed along with schema number. Now enter the field name and field type for insertion onto the data base, along with a "Set" type of S, A, Q, J or H. S denotes Single Details, A = Address Details, Q = Qualification Details, J = Job Details and H = Absence/Lateness History.

ALLOWABLE FIELD TYPES ARE:

X30	- 30 character alpha field
X10	- 10 character alpha field
D	- Date field
Q	- Six digit quantity (signed)
V	- Value - signed - to 6 pounds and 2 pence.
T	- Telephone Number or 10 digit numeric.

7. PAYPLAN DETAILS (again from DEFINE PARAMETERS)

PAYPLAN is the Coggon Computers Payroll System written for the HP3000.

This screen sets the rules for the transfer of information from PAYPLAN into PERSON.

All PAYPLAN items entered via this parameter screen will be regarded as "transfer" items during the day by day processing of PAYPLAN. If any of these transfer items are changed within PAYPLAN by means of the Employee File Maintenance program a record of this change is stored in the PAYPLAN TRANSFER FILE.

Then at any time within the processing of PERSON, by suitable choice of function key, this transfer file can be processed, resulting in these changes taking effect on the PERSON file.

These changes take effect either completely automatically or with display and possible further input, according to choice of Transfer Type (see below).

Enter:

PAYPLAN Reference	The data number within PAYPLAN of the field to be transferred.
Data Dictionary Number	Give it a data dictionary number within PERSON.
Transfer Type	Enter A for automatic non-display; P for automatic, display and prompt.
Prompt Message	Up to 20 character message to appear on the PAYPLAN PROMPT screen.
Field Name	Up to 20 character field name for labelling purposes.
Schema Number	Schema number within PERSON.
Field Type	Field type within PERSON as described in 6 above.

Now up to ten further fields can be defined against the same PAYPLAN reference, with an associated prompt message. Fields so defined will be prompted on the PERSON screen (PAYPLAN PROMPTS) and can be entered during the processing of the PAYPLAN TRANSFER FILE. Thus information collected in its basic form during PAYPLAN (eg. Leaver this Period) can be supplemented by further background details (eg. Date of Leaving, Reason for Leaving) during PERSON.

8. MAINTAIN FILES - screens of maintenance and enquiry

Examples of these screens contain:

Personal Information	name, phone number etc.
Associated Address Details	Such as Doctors N/A, Next of Kin N/A, Wife and Children.
Educational qualifications	repeated for each Educational Establishment, Examination and Subject as required.
Employment Information	showing job titles, grades, dates salaries for each job.
Absence History	maintaining a record of each occurence of absence or lateness, showing SSP relevant information including notification dates, and reasons for absence. Records of warnings given are also maintained using this screen.

9. USER SCREEN (Maintenance/Enquiry)

This screen enables the relevant User Screen to be chosen and should not be confused with definition of User Screen - discussed later in paragraph 12.

Selection of USER SCREEN within file maintenance gives an index consisting of a screenful of Screen Numbers against Screen Descriptions associated with that operator code.

Select the relevant Screen Number, enter in NEXT and that User Screen will be displayed for maintenance.

NEXT SCREEN and PREVIOUS SCREEN enable easy movement throughout the index.

Selection of TOTAL INDEX will replace the information displayed by the whole index, for all operators. OPERATOR INDEX returns to the specific index for that operator.

PERSONAL DETAILS returns to the initial Single Fields screen for that employee.

10. PAYPLAN PROMPTS

Selecting PAYPLAN PROMPTS automatically processes the "Transfer File" if applicable. For each employee that has had a change within Employee File Maintenance of PAYPLAN during the period, a screen is displayed within PERSON provided that the field changed has been defined as transfer type P on the PAYPLAN DETAILS screen.

The Employee Code and name are displayed and then each of the PAYPLAN fields showing the previous contents of the field and the field after amendment.

Then a series of prompts are displayed, as entered within the Payplan Prompt Definition Screen, and the entry of further fields of information are enabled. This is reported for all type "P" transfer items applicable to that employee.

All employees affected are dealt with in this way until the transfer file is exhausted.

NOTES

1. Transfer fields defined as type "A" will not appear on this screen.
2. All fields changed (type "P" AND TYPE "A") will result in an audit trail record and will thus appear in the transaction report.

11. INFORM

All of the screens available in MAINTAIN mode are now available for use in an INFORM mode only.

They are presented in an identical format to their MAINTAIN screen counterparts with all file amendment facilities within the screens and update relevant function keys removed.

NOTE: A User Screen can be established as file maintenance screen and will always operate as such using the MAINTAIN FILE route. However, if it is accessed via INFORM then it will operate as a screen of information only, the AMEND function keys becoming inhibited.

12. "USER SCREEN" FACILITY

Facility is given for each User to define and label his own personnel screen which enables access/update of just those fields which are of interest to that User.

When USER SCREEN has been selected the next available screen number is allocated and an entry is expected, to associate a name with the screen. Alternatively an existing screen number can be entered to display for amendment the existing USER SCREEN.

Facility is also allowed to generate a new screen using an existing screen as a base.

Now, at the foot of the screen enter:

Row No - Row number on the screen (2-23)

Data No - Data dictionary number (if known, see below)

Description - The Data Dictionary descripiton will be suggested and can be amended if Data Number is entered. Otherwise Data Dictionery Number entered will be used to locate the data item - see note 3.

Label Position - Column number of the label (1-80)

Data Position - Column number of the data item

Display Type - H, F, B to denote half bright, full bright, blinking.

Access Type - A, R, W or blank to denote action code and key item, read only, write only, or both read and write.

Select - Can be labelled 1, 2 or 3 against any item

As each data item is entered in this way, the body of the screen is built up in the rows and columns defined.

If "Select" is indicated (by a label 1, 2 or 3 to refer to the selection) then a further line of information is invited to define the selection.

Enter:

Select Item - Data Dictionary Number
Operation - Only >, <, =, NOT>, NOT<, NOT= are allowed
Criterion - Enter constant for comparison
Select Code - Select code 1, 2 or 3 under definition.

NOTES:

1. Reference can be made in "data no" to any item from the Tables defined in the Set Tables Screen. This is accomplished by entering a letter to denote the table followed by two digit code.

2. The function keys associated with the screen generated will be fixed:

   FK2 - NEXT EMPLOYEE - display next employee
   FK3 - ENTER - allow data to be changed
   FK4 - UPDATE - update with altered data
   FK7 - HELP
   FK8 - EXIT

3. The first ten characters of data item description are used to locate the data item in the absence of Data Dictionary Number in previous field. If this results in a non-unique data item, a message to this effect is displayed and cycling through the possibilities is enabled.

13. DEFINE REPORTS

Facility is given for each User to define and label his own personnel reports in the same way as USER SCREEN can be defined.

The next available report number is allocated and an entry of Report Name is expected, to associate the report with the number. Alternatively an existing report number can be entered to display for amendment the associated "define report" parameters.

Enter:

Line No - Items with the same line number appear on same line.
Data No - Data dictionary number (or zero for label only).
Label - The schema name will be suggested but this can be amended.
Label Position - Print position of the label (1-132)
Data Position - Print position of the data item
Select - Can be labelled 1-6 against any item.
Summary - If marked S - detail item does not print - Summary only.

These details are entered for each print heading or print line.

When all items have been defined in this way, the selection, sequence and up to three calculations are now defined:

Selection:

Select Item - Data dictionary number
Operation -), (, =, NOT), NOT(, NOT= as before
Criterion - For comparison
Select Code - 1-6 to represent the select item being defined

Sequence:

Up to 4 data dictionary numbers from major to minor sequence.

Calculate:

Up to three calculations can be defined C1, C2, C3 by linking data dictionary numbers (or data) with operations. Allowable operations are:

+	add
-	subtract
*	multiply
/	divide

NOTES

1. Data number can contain a data dictionary number, or a table reference (letter + 2 digit code) or C1, C2 or C3 denoting a defined calculation.

2. Literal - L followed by the literal.

14. CONCLUSION

The idea of this presentation is to establish the concept of PERSON, rather than detail all of the contents of the application itself.

There are further reports and processes - as would be expected of a Personnel Information System:

e.g.	General Update	is an update that can be run at any time and after a question and answer session to determine which items should change, (whether to remove old leavers, change salaries), will then read through the Data Base to perform the updates. This is also used to perform WHAT IF requests.
	Monthly and Quarterly Manpower Reports	statistics on Male, Female, Part Time, Full Time employees with Labour Turnover Percentages.
	udit Trails	of changes made.

The important message is that the system, developed in this way is:

1. Completely Flexible — enabling Personnel Officers to get all items of information PRESENTED IN THE WAY THEY REQUIRE IT - subject only to having relevant security level clearance;

and surprisingly:

2. **Completely Understandable** — When the system had been defined we felt we had an application that was comprehensive but User Complicated. It was a pleasant surprise to discover that Personnel Officers very soon came to terms with the concepts of User Screens and User Reports and are happy with the operation of PERSON.

June 1983

A COMPARISON OF THE OSI REFERENCE MODEL & HEWLETT-PACKARD'S DS3000 TELECOMMUNICATIONS SOFTWARE

BY: Gregory J. Sannik
Lonza Inc.
22-10 Route 208
Fair Lawn, NJ 07410

FIGURE 1
SIDE BY SIDE COMPARISON OF OSI AND DSN(DS3000)

LAYER	OSI MODEL	DSN	LAYER
7	APPLICATION LAYER	USER LAYER	5
6	PRESENTATION LAYER		
5	SESSION LAYER		
4	TRANSPORT LAYER	NETWORK ACCESS METHOD LAYER	4
3	NETWORK LAYER	COMMUNICATIONS MANAGEMENT LAYER	3
2	DATA LINK LAYER	COMMUNICATION ACCESS METHOD LAYER	2
1	PHYSICAL LAYER	COMMUNICATIONS LINE CONTROL LAYER	1

BACKGROUND OF OSI AND DSN

The Open Systems Interconnection was first proposed in 1978 by the International Standards Organization as a means of developing compatible communications protocols between heterogenous vendor hardware. The OSI reference model specifies a 7 layer strucure; each layer discretely interacting with the layer(s) adjacent to it. Specifically, this interaction is such that services are provided to layer n by layer n-1 in response to layer n requests. At opposite ends of the model, its lowest layer provides service by physically transmitting data in bits over a medium known as a "communication channel", while its uppermost layer makes requests for interface connection/release between two or more user processes. The layers in between bridge this gap between the very complex request made at the top of the model and the primititive, single-purpose task provided by the bottom of the model by utilizing an approach in conformance with the philosphy of "modular design"; each layer performing a specific, well defined task, interacting with adjacent layers using a precise dialogue of requests, indications, responses, and confirmations.

DSN was first announced by Hewlett-Packard in 1975 as an upgrade/replacement to their Network Central Communications Package and Network Satellite Communications Package. DSN provides a peer to peer relationship between nodes of a distributed processing network.

PURPOSE OF PAPER

The purpose of this paper is to compare a product in the DSN line, DS3000 (used by the vendor's Model 3000 processors), with the OSI model. As is the case of other popular network implementations, DS3000 is not a clone of the reference model. Although there, are fundamental similarities between the two, important differences exist with regard to implementation specifics. These differences are particularly profound at the higher layers of DS3000 and OSI.

DSN(DS3000)'s FIT IN THE OSI FRAMEWORK

Like OSI, DS3000 possesses a layer structure. However instead of 7 layers, DS3000 is structured on 5 separate functional layers. (SEE FIGURE 1.) Layer 1, the Communications Line Control (CLC) conforms to guidelines set forth as part of OSI's Physical Layer; both providing a physical link between adjacent network nodes via a transmission media. Various interface standards are supported by the CLC. Included for short distance interface are; RS-232-C (DSN/INP) and RS-442 (Direct Control Link). Long distance data link protocols are also supported (e.g. V.35, X.25, and X.21) using modems.

Layer 2 of DS3000 is called the Communications Access Method (CAM) Layer. The CAM layer is quite similar to the data link layer of OSI in that both provide data transfer over the physical medium with the necessary synchronization, error control, and flow control functions. As stated earlier, DS3000 supports an array of data link protocols to accomodate interface to Public Data Networks and for direct, node to node interface.

Direct node to node links are established, maintained, & severed using a modified bisynchronous protocol that responds to text transmission with text acknowledgement. Transmitted text is segregated into text blocks. Each block contains a 16 byte header which specifies a process i.d., message type, and length, and a data compression/non-compression flag. In addition to the 16 byte header, an optional appendage section (a header extention) may also be present.

Among the long haul protocols supported by DS3000 is X.25. Specific DS3000 implementation of X.25 entails an X.21bis (RS-232-C, CCITT V.24) interface, LAP-B protocol with window sizes of 1 to 7 bits, and switched or permanent virtual circuits with window sizes from 128 to 1025 bytes. High throughput datagram services are still yet to be supported by DS3000. Also noteworthy is the fact that the X.25 protocol has been found to yield greater throughput than HP's Bisync. The cost though, is a greater CPU load to handle the improved throughput.

CAM error checking employs longitudinal, diagonal, or vertical parity checks. Retransmission is initiated automatically upon error detection.

Layer 3 of DS3000, the Communications Management (CM) Layer, like the Network Layer of OSI provides communications routing, flow control & lock-up prevention as well as error detection. CM architecture supports most of the more common routing topologies such as the ring, string, and star topologies. CM routing is non-adaptive and is accomplished through a nodal addressing scheme. Tables of nodal addresses establish routing paths on which messages are passed. All networking functions are handled on a point to point basis, between locally or remotely adjacent processors. Since data packets do not contain addressing information, routing must be specified programmatically or with FILE commands that establish logical/ physical links between adjacent nodes.

The fourth layer of DS3000, the Network Access Method (NAM) Layer like the Transport Layer of OSI provides the bridge between application functions of the higher layer(s) and the transmission functions of the communications subnets of OSI and DS3000. This bridge creates a virtual end to end interface between peer entities. In providing access to the DS3000 subnet, the NAM, like the OSI Transport Layer, "packetizes" messages in preparation for transmission. When a message is to be transmitted Network Service Intrinsics software within the NAM proccesses out-going messages. Incoming messages are processed by the NAM's Network Interface Monitor software and are linked to user programs.

With regard to NAM upgrade, Hewlett-Packard late last year announced that they plan to incorporate OSI's Class 4 Transport Layer Specification into their telecommunications package. The class 4 specification unlike the class 1-3 specifications provides detection of damaged, out-of-sequence, or lost packets, a prerequsite for support of datagram service as well as local area network applications.

The fifth and final layer of DS3000 is the User Layer. It is this layer that provides direct services to the user. In this regard DS3000's User Layer is congruent with the Application Layer of OSI. Functions that the OSI model's Application Layer specify fall into three categories of service elements; common, application, and user specific. As defined in OSI, each category provides services of varying specificity. A common service element is "common" to all applications (e.g. establishing an interface between the Application Layer and the lower layers of the model). Application service elements are related to a general transaction type (e.g. virtual terminal access, public data network access, network file transfer, etc.) while user service elements are related to a particular mix of transaction types that is used to fulfill a specific user requirement (e.g. real-time inventory tracking in a multi-plant environment, air line reservation system, etc.).

Among the application specific functions provided by DS3000 are virtual terminal access, network file acess, public data network acess, network file transfer (i.e. DSCOPY), network peripheral access, remote IMAGE data base access, and network interprogram communication.

Although DS3000's User Layer is quite similar to OSI's Application Layer to the extent that it provides functions specified by the model, it is not a carbon copy of Layer 7 of the OSI model. Specifically, the User Layer of DS3000 seems to provide other services that exceed OSI Applications Layer specifications and that are instead addressed by the Presentation & Session Layers of OSI. These services are provided on an as requested basis at the time when the virtual circuit is established via parameters of the DSCONTROL, DSLINE, & REMOTE HELLO commands.

The OSI Presentation Layer provides any data transformations required to increase data throughput (e.g. compression), ensures data security (encryption), and/or enables interface between disimilar peripherals (in particular terminals) that require conversion of character sets, scroll to page (or page to scroll) display , and/or cursor addressing. The DS3000 User Layer provides some but not all of these services. Data compression is provided through the COMP parameter of the DSLINE & DSCONTROL commands. Scroll to page conversions require use of a buffer size parameter; LINEBUF = n; where n is the number of words the DS3000's line buffer can handle without truncating or wrapping. Data encryption however is not supported by DS3000.

The Session Layer of OSI is said to bind the upper layer entities and the application process into a logical communicating relationship. Specific services provided by the Session Layer include; session connection establishment and release, normal data exchange, quarantine service, expedited data exchange, interaction management, session connection synchronization exception reporting.

DS3000's User Layer supports several of these functions. Expedited data transfer is achieved through the EXCLUSIVE parameter. Its use gives the virtual circuit exclusive access to the physical circuit. Session connection establishment, management, and release is facilitated through use of REMOTE HELLO & REMOTE BYE Commands. Use of these commands though will leave the virtual circuit intact. Exception reporting is handled by a TRACE facility that provides a record of line actions, states, and events that occur during DS3000 operations. DS3000 interaction management capability can be used to set up the local processor as a "slave", "master" or "master/slave" node.

FINAL OBSERVATIONS

User response to the OSI Reference Model has been overwhelmingly positive. So strong has been their response, that so called "industry experts" have called the phenomenon a "user rebellion". Vendor's were reluctant at first to conform to the model. However, gradually the major ones have begun to incorporate the basic principles of OSI into their network implementations. Hewlett-Packard, recognizing the importance of the model and user support for it, has also come out to endorse it. In a joint news release made late in 1983 with 20 other U.S., European & Japanese companies, HP endorsed layers 1-3 of OSI. It is expected therefore that HP will continue to support the OSI concept through further upgrades/modifications of their DSN product line.

REFERENCES

Brawn, Mel and Hibbard, Carol. Selection Criteria to Choose Bisync or X.25 Protocols for use with DSN/DS in HP3000 IUG Conference Proceedings, Montreal Canada. 1983 Vol 1, pp44-1:13.

DS/3000 Reference Manual. Cupertino, CA: Hewlett-Packard Company, 1980.

Hewlett-Packard Distributed Systems Network (HP-DSN), Datapro's Report on Mini-Comnputers, Delran, NJ: Datapro Research Corp., January 1983, pps. C11-472-101:107.

HP3000 Computer Systems: General Information Manual. Cupertino, CA: Hewlett-Packard Company, 1980, pp 4-3:12.

Interact. San Altos, CA: Hewlett-Packard's International Users Group, Sept/Oct 82' Vol 2, Issue 5, p19.

Meier, Edwin E. High-Level Protocols, Standards, and the OSI Reference Model, Data Communications. New York: McGraw-Hill Publications Company, July 1982, pps 71-101.

Open Systems Interconnection Reference Model. Vienna, VA: OMNICOM Information Service, June 1982.

Tannenbaum, Andrew S. Computer Networks. Englewood Cliffs, N.J.: Prentice-Hall Inc., 1981.

DISTRIBUTED PROCESSING
IN AN IBM-HP ENVIRONMENT

by Rolf Frydenberg
Project Manager
Fjerndata
Norway

CONTENTS

1. Abstract
2. Introduction
3. Software for HP3000 to IBM communication
4. Remote Data Access - in Batch Mode
5. Remote Data Access - Interactively
6. Simultaneous access to IBM and HP
7. Concluding Remarks

1. Abstract

This paper will present an overview of different approaches to distributed processing, where both IBM and HP computers are involved. The advantages and disadvantages of the different methods or approaches will be discussed.

We will discuss the available datacommunication software for accessing IBM mainframes from HP3000s, and suggest some methods by which the user can supplement the inherent functions of these software products.

We will concentrate on interactive distributed applications, where access to IBM programs and data is the central theme.

2. Introduction

There are many approaches to implementing distributes applications. Below we will describe these according to what we have chosen to call their level of sophistication. We have found five basic levels of sophistication that are possible when application are distributed between HP3000s and IBM mainframes. Below is a presentation of these levels, as seen from the perspective of the HP3000 user.

A) Batch-only access

This is the most common situation, where a user submits a batch job through RJE or MRJE and retrieves output. The output may be directed to a printer or a file.

B) HP on-line, IBM batch

Many data-entry applications work in this manner. Transaction programs are run on the HP3000 that enter data into files. These files are transmitted to the mainframe as batch jobs, where they update a central (corporate) database.

C) HP Passthrough, IBM on-line

The HP3000 emulates an IBM3270-type cluster controller, and an attached terminal emulates an IBM3270-terminal, thereby allowing a user access to both IBM and HP3000 applications from the same terminal, but at different times.

D) HP batch, IBM on-line

A batch job on the HP3000 retrieves data (to a file or an application) by accessing on-line systems on the mainframe.

E) Simultaneous on-line

A user runs a program on the HP3000, that accesses the HP3000 or the mainframe, as the need arises. Data may be retrieved from, or entered into, HP3000 or mainframe-applications at will.

In most cases, level E would probably be acknowledged as the "ideal" that one should try to achieve. This is possible with DSN/DS between HP3000s. Of course, DS can also achieve the 4 less sophisticated levels as well. For access to IBM mainframes, levels A and B are the most common, using DSN/RJE or DSN/MRJE. The next three levels are possible by using DSN/IMF. Currently, most IMF-users only utilize level C - the PASSTHRU program in DSN/IMF or IMAS/3000.

3. Software for HP3000 to IBM communication

It is an established company policy of Hewlett Packard's to provide for three basic types of compatibility within its datacommunications strategy: Between HP computers, with internationally standardized networks and carriers, and to IBM mainframes, through IBM's System Network Architecture (SNA).

There are three HP datacommunication products for access to IBM mainframes: DSN/RJE, DSN/MRJE and DSN/IMF. The common prefix for these product names - DSN - is an accronym for Distributed Systems Network. The same prefix is used for a large number of other HP datacomm. products:. DSN/DS, DSN/INP, DSN/ATP, etc. The product accronyms mean Remote Job Entry, Multileaving Remote Job Entry and Interactive Mainframe Facility, respectively. In this chapter we will look a little closer at these products.

DSN/RJE and DSN/MRJE have been around for a number of years, and are much-used systems that are well thought of by most users. DSN/IMF is relatively new (announced as IML in 1980, and rewritten and renamed in 1981). So far, even though the interest among users is high, the number of installations is still very low compared to MRJE and RJE.

3.1. DSN/RJE

This is HP's 2780/3780 emulator. RJE gives the user batch access to IBM mainframes and other types of computers. It uses a one-file-at-a-time type of protocol.

RJE emulates a "dumb" communications processor, with essentially 1 printer, 1 card punch and 1 card reader. RJE is an operator-oriented communications systems, i. e. an operator must issue commands - which files to transmit, where to route the received print or punch files, etc.. Only one user can use RJE at a time.

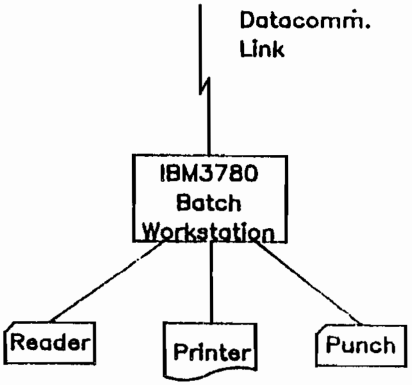


Fig. 3.1: A 3780 device

RJE's major advantages are that it is very unsophisticated (and therefore not particularly error-prone); and that it implements a well-established standard (thereby giving the user compatibility with a large number of different computer systems). The disadvantages, though, are generally considered greater than the advantages. Most significant are the unsophisticated user-interface, and the filesize and -format limitations.

3.2. DSN/MRJE

This is HP's Hasp Work Station emulator. providing more flexible and automated batch connection to IBM mainframes. It provides for transfer of multiple files, to mutiple devices, simultaneously.

MRJE can emulate up to 7 printers, 7 card readers and 7 card punches simultaneously. MRJE does not require an operator for normal operation. Any user may transmit files through the MRJE user interface program, and determine the output-device or output-file for his own job.

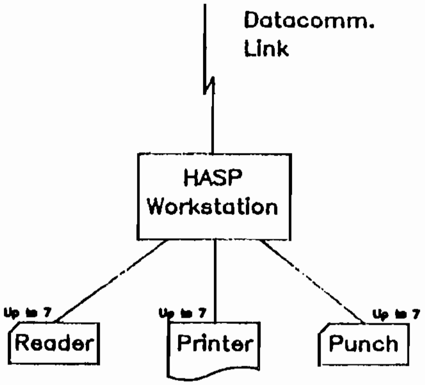

Datacomm.
Link
HASP
Workstation
Up to 7
Up to 7
Up to 7
Reader
Printer
Punch

Fig. 3.2: A HASP work station

MRJE's major advantages are the user-interface program, allowing any user to submit jobs to the mainframe, whether the communications link is up or not; and the ability to receive print- and punch-files without operator intervention.

3.3. DSN/IMF

This is HP's 3270 emulator, providing interactive ("passthrough") access to IBM mainframes, or program-to-program communication. It works with BSC or SDLC line protocol.

IMF is a multi-component software product. Seen from the user's point of view, it consists of two basic components: The Intrinsics (for programmatic access) and PASSTHRU (for emulation of IBM327X display terminals and IBM328X printing terminals). Other basic components are the communications driver, the cluster controller emulator, and the management program.

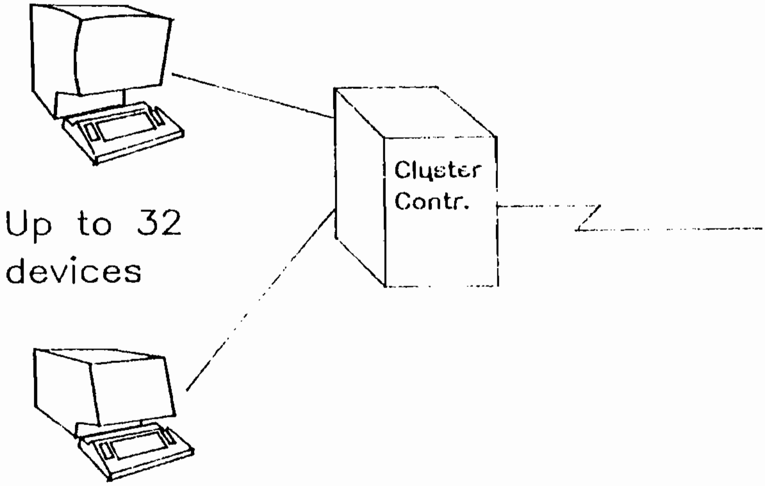

Cluster
Contr.
Up to 32
devices

Fig. 3.3: An IBM 3270 cluster

4. Remote data access - in batch mode

Batch mode access to remote data means that the data access seen from the HP3000 user occurs in batch mode. This can be done through batch communications systems such as DSN/RJE or DSN/MRJE, or by accessing the mainframe as an interactive user, but from a batch job running on the HP3000.

The batch communications protocols for access to IBM mainframes, 2780/3780 and HASP Work Station, are primarily oriented to submitting a deck of punch-cards, in the form of a "JOB" to the mainframe, and retrieving output to a card punch or a line printer. The main difference between a 2780/3780 an HASP Work Station is the number of devices (card readers, card punches and line printers) that are supported.

Normally, RJE (the 2780/3780 emulator on the HP3000) and MRJE (the HASP Work Station emulator) are not thought of as components in a distributed application. But there are a number of options open to users in order to put more intelligence into the usage of these products. In Example A, below, we will look at one approach to integrating MRJE into the distributed applications of a user.

4.1. Example: Spoolfile Distribution

The problem:	Spoolfiles are received at the HP3000 from the centrally located IBM mainframe. Many of the files received are not supposed to be printed here, but at remote spooled printers.
The solution:	This problem can be solved by looking up the IBM job-name and FORMS-code in a table , and then redirecting the spoolfiles according to the entries in this table.

First, let us look at how the operators "fixed" the problem using standard commands:

All the remote spooled printers - at this site - have a unique device class name, generally with the form LPxx, where xx is the logical device number of the printer. The mainframe users were instructed to use this device name as the IBM FORMS-code for the listings that they wanted to have transferred to their remote spooled printers.

Every time a forms request from MRJE with FORMS-code LPxx appeared at the console, the operator refused printing the spoolfile (REPLY pin,N) and redirected it to the remote spooled printer LPxx (ALTSPOOLFILE #Onnn, DEV=LPxx).

This of course, works. But the solution soon became a new problem: The users found that having their spoolfiles printed at their own printers was very nice, and so they generated more spoolfiles for their own printers, and more and more, and ..... Very soon the operators had little else to do than redirect spoolfiles. That was when we were asked to look at the problem.

The answer we found was quite straight-forward: To implement an "automatic operator". Our implementation is the following:

A batch program runs SPOOK (HP's standard spoolfile utility program) as a SON process passing data through message-files. The following commands are sent by the program to SPOOK:

SHOW	To retrieve status information on the spoolfiles, and to find out which files are candidates for redirection.
TEXT & LIST	To get hold of the IBM FORMS-code.
ALTER	To change the device class for the "current" spoolfile.

The program decides where to send a spoolfile based on two criteria - the IBM jobname (which by MRJE is converted to a filename) and the IBM FORMS-code. This information is contained in a file which is read when the program starts execution, usually at system start-up time.

4.2. Example: A distributed memo-switching system

The problem:	The company has message switching systems on both mainframes and HP3000s, e.g. MEMO or PROFS on the IBM mainframe, JENNY or HPMAIL on HP3000. The users do not want to log on to both systems in order to retrieve memos.
The solution:	A batch job accesses the mainframe, searches for memos to HP3000 users, copies these to local files, and then places them into JENNY/HPMAIL, and vice-versa.

With the advent of the automated office, electronic message switching is becoming increasingly more popular. On the HP3000 serveral systems are already on the market. These include HP's HPMAIL and Infomedia's JENNY. A large number of message switching systems is available on IBM mainframes as well.

Users with both IBM mainframes and HP3000s have several options available for integrating mainframe and HP3000 message switching systems:

Users can access the mainframe via the pass-through mode in DSN/IMF or IMAS, and read and create messages to other mainframe users on-line.

If the mainframe message switching system has a batch access facility, a JOB may be submitted regularly from the HP3000 (through RJE or MRJE) that can enter messages into the mainframe system and retrieve messages from it.

A special program may be written to access the message switching system on both computers, this program will then transfer messages between the systems.

For our own implementation, we selected the last of these possibilities. This is the most flexible one, and lets every user use the system with which he is most familiar, whether that is te mainframe or the HP3000.

This solution would have been very difficult to implement using standard DSN/IMF intrinsics, we estimate approximately 3000 - 4000 statements. By using the Automatic Dialogue Facility in IMAS, we could get away with about 200 Autodialogue statements for the IBM access part of it. The rest is done through TDP USE-files and standard JENNY/HPMAIL commands.

The IMAS Autodialogue facility is a very high level special-purpose programming language for accessing IBM mainframes. Autodialogues are interpreted by IMAS, running interactively or in batch mode. for more information on what Autodialogues are, consult the Reference Manual for Fjerndata's IMAS system.

5. Remote data access - interactively

Data on the remote mainframe may be accessed interactively in two "modes": passthrough mode, where the user uses his HP-terminal to emulate an IBM3270-terminal; or in programmatic mode, where the user controls data transfer interactively, by using functions such as IMAS automatic dialogues.

5.1. Passthrough mode

In passthrough-mode, the HP3000 emulates an IBM3270 cluster controller, and HP-terminals are used to emulate 3270-type screen terminals or 3280-type printing terminals.

Before we go on to describe the two programs that can do passthrough-mode interactive access, let us look at the reasons why users want this type of capability.

One reason, of course, is economical: If you have on-line applications on the IBM mainframe as well as on the HP3000, and the same users need to access applications on both computers, you do not want them to require two terminals to do this.

Another reason, is standardization: If you want to have a single supplier of terminals, then you need passthrough-mode on the HP3000, since IBM does not supply passthrough-mode access to HP3000s. (But you could ask for it, of course.)

A third reason, is the datacommunications network: Asynchronous HP-terminals cannot easily share a datacommunications line with synchronous IBM-terminals. For screen terminals, DSN/MTS (HP's multipoint BSC protocol) might be used, but it is a costly solution, and is not compatible with IBM's SDLC protocol.

For one or several of the above reasons, and possibly some others as well, you have reached the conclusion that you want passthrough-mode interactive access to IBM mainframes from your HP3000s. The next question is: What are the alternatives?

Well, there are two alternatives: DSN/IMF comes complete with a program called PASSTHRU, which can emulate IBM3270 and 3280 terminals, and Fjerndata has a product called IMAS, which uses DSN/IMF, and can emulate IBM3270 terminals.

Since the main area of overlap between PASSTHRU and IMAS is in passthrough-mode emulation of IBM3270-terminals, let us briefly look at the differences, and how they affect performance and end-user productivity.

5.2. Programmatic mode

In programmatic mode, applications, files and databases on the IBM mainframe are accessed through programs. Here, as in pass-through mode, are two alternatives:

> DSN/IMF includes a set of INTRINSICs that allow programs written in most programming languages (from SPL to BASIC) to access the screen or printer images received from the mainframe.
>
> IMAS has a very high level language built into it, called the IMAS Automatic Dialogue Facility. Using IMAS Autodialogues screeen images are more easily accessed than through the DSN/IMF intrinsics, and the user has control while the autodialogue executes.

Using the DSN/IMF Intrinsics is not as simple as most other types of programming. Just a simple application that logs on to the IBM mainframe executes one command and displays the received screen on a terminal would probably require something of the order of 1000 statements. A "real" application needs a lot more, including procedures for error-handling and other utility type functions. IMAS, which is a program that uses the DSN/IMF intrinsics, consists of more than 10.000 statements.

Using IMAS Autodialogues, though, is quite simple. An autodialogue that logs the user on to an application, going through 3-5 screens to do so, typically does not require more than about 50 Autodialogue statements - including error-handling.

Autodialogues are used for a variety of other purposes than logon/logoff. The distributed message switching system described in the preceding chapter was implemented using the IMAS Autodialogue facility, using about 200 statements for logon, logoff and file transfers.

FUNCTION	PASSTHRU	IMAS
Terminal handling	BLOCK MODE	CHARACTER MODE, mod. fields
Function keys	8 Hard-coded function keys	4 userdefinable keysets
Terminals handled	HP Block mode terminals only	All HP terminals
DSN/MTS support	YES - block mode	Only through cluster contr.
Printer support	External print-device only	Internal printer supported
Access to commands: and programs	Must EXIT	MPE MODE
Advanced functions	none	Automatic dialogues
Batch access	Only through INTRINSICS	Automatic dialogues

Figure 5.1: A Summary of differences between PASSTHRU and IMAS

5.3 Simplified access to the mainframe

Most HP3000 users know a little about using the HP3000 - what tasks they can do and how to go about it. When given access to IBM mainframes too, they will have to learn how to use this computer as well. And most people with both HP3000 and IBM mainframe experience, will tell you that it is a lot more difficult to learn how to use the latter compared to the former.

To get around this problem, and to help the user pass the knowledge-threshold to IBM mainframes, IMAS includes facilities for customizing the product to how the user wants to access the IBM mainframe. There are two possible ways to do this customization:

A) IMAS configuration file:

The configuration file may be specific for the individual user, or for a group of users. The configuration file is backreferenced to a :FILE-equation, making it extremely easy to implement account- and system-wide defaults through the MPE User Defined Command facility.

Parameters that are defined in the configuration file include: Which IMF configuration file to use, which of the emulated IBM3270-terminals to open, what to trace when the trace facility is invoked, when to use autodialogues, and what functions and labels are in which key-set.

IBM 3278 screen terminals have 24 Program Function keys (PF1 through PF24), three Program Attention keys (PA1, PA2, PA3), and an host of other special-function keys (Erase End of Field, Erase End Of Input, CLEAR, RESET, SYS REQUEST, etc.). HP262X terminals have only 8 user-definable function keys (f1 through f8), and a limited number of fixed-function keys (clear line, clear-display, roll-down, roll-up, etc.). These differences between IBM- and HP-terminals imply that a direct mapping between HP- and IBM-function keys is infeasible.

In IMAS we have allocated a two-digit function number (from -3 to 99) to each IBM or local function that can be performed. A list of these functions may be found in Appendix A. These function numbers are assigned in groups of eight (a "key-set"), to the eight function keys on the HP262X terminal. There are also special function numbers for switching from one key-set to another. For each defined function, a label may also be defined. This label is displayed by IMAS in the special label-field of HP262X terminals (lines 25 and 26 on the screen).

B) Automatic Dialogue Facility

When using on-line applications on the mainframe, the user must often go through a number of transactions in order to get into the screen that s/he really wants to enter data into. Logging off the same application may be equally complicated. For this reason, the IMAS automatic dialogue facility was invented. An automatic dialogue is a description of a set of interactions between a user and a mainframe application that is executed by IMAS. Operations that are performed repeatedly will not have to be done by users, but may be performed by the automatic dialogue facility, removing some of the tediousness from the terminal-operator's work.

Automatic dialogues can also access files on the HP3000, and in effect perform data transfer operations between on-line mainframe applications and MPE files. Automatic dialogues may also be executed in batch mode.

6. Simultaneous access to IBM and HP

This is the realm of distributed databases in practice, where the programs access the data wherever it is - whether that is on the local HP3000, a remote HP3000 connected through DSN/DS, or on an IBM mainframe connected via DSN/IMF. This is the type of application that DSN/IMF was developed to support.

From the HP3000, one does not have direct access to any of the mainframe Data Base Management Systems. You cannot execute DL/1 calls directly but must view the mainframe database through an on-line application on the mainframe.

The sets of screens that the mainframe application exists of may be viewed as a database, of the "hierarchical" type. In figure 6.1, below, we have illustrated this.

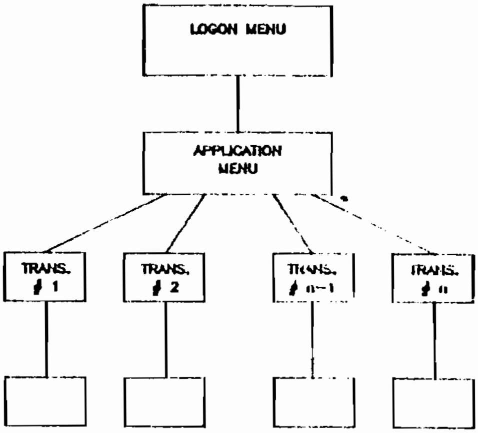


Figure 6.1: A "Database of Screens"

This figure illustrates a typical mainframe application: First, you must logon to the application of your choice, then you select the transaction that you want to use, and finally you use this transaction, which may consist of several screens, to enter or retrieve data.

Comparing figure 6.1 to a database, the log-on menu is used to issue a request for a specific database, the "application menu" is used to select a DATASET and the transactions are used for GET, PUT, UPDATE, DELETE operations against the entries in the dataset.

Let me illustrate this approach a little more in detail.

At our installation (IBM 3081K & 370/168 & 4341-2) we run the IBM operating systems VM (on the 370) and MVS (on the other two processors). Datacommunication is handled through VTAM. We run systems like CICS (w/MANTIS), TSO (w/APL, SPF, IFPS and GDDM), IMS DB/DC, and a host of other subsystems. A user who wants to log on to an application under the IMS database system, would go through the following steps:

1. Turn on his terminal (from the 3000 this means he starts the IMAS program). This would cause our VTAM "welcome-message" to be displayed on the user's screen.

2. Enter the IMS subsystem by issuing the IMS command. This would get him two screens in succession. The first is a message like "OK, LOGON ACCEPTED", then a message from IMS saying "Terminal Connected".

3. The user will now log on to his application by accessing a format. This means inputting for example "/FOR abcd", where abcd is the name of a defined format. This would be the logon screen for the application the user wants to use.

4. The user gets a screen that prompts him for some kind of user id and password, and possibly some type of billing info (acct. no., proj. no. etc.). The user inputs this data and hits enter. He now gets the main menu for the "abcd" application.

Any approach to programmatic access, would have to go through the same steps. In the IMAS autodialogue approach, these steps are simplified. The user would typically enter a command (UDC), such as ABCD, and this would get him all the way into the primary menu of application ABCD on the mainframe. For security reasons, the user should probably be prompted for his password, at some step in the dialogue. This is done quite easily.

7. Concluding Remarks

In this paper we have presented three HP software-products for access to IBM mainframes. We have shown what these products can do, and how other applications can enhance the former ones..

Our recommendation to other users is to use MRSE for batch access, and to use the IMAS/IMF combination for online access.

Our own experience with distributed databases is quite limited, but generally these databases cost more that they are worth. Using approaches such as the file copy utilities in IMAS, to copy files between HP and IBM is probably the best approach.

For installations with many remote spooled printers, that all want to receive output from the mainframe, using a system like SDS makes a lot of sense.

THE COMPUTER USER/NETWORK INTERFACE
AT GLAXO PHARMACEUTICALS LIMITED

Contents

1. Introduction to Glaxo

2. Evolution of computing in Glaxo Pharmaceuticals

3. The HP3000 Network

4. The User Interface and other concerns

5. Glaxo Pharmaceuticals User/Network Interface

6. Summary of benefits

1. INTRODUCTION TO GLAXO

Glaxo is one of the world's leading and most successful healthcare companies, with an outstanding research record. It employs almost 30,000 people worldwide, many of them graduates in scientific disciplines, and its products are sold in virtually every country in the world. Glaxo has a market value of over US $5,500 million and annual sales in excess of US $1,400 million.

Glaxo is also the only British group in the world's top 20 pharmaceutical companies and is one of the fastest growing. The Group has subsidiary companies in over 50 countries many with their own manufacturing facilities, and agency representation in more than 100 others.

In the UK, Glaxo Pharmaceuticals Ltd is responsible for the manufacture of finished products for sale both in the UK and overseas also providing extensive marketing and technical services to operating companies throughout the world.

2. EVOLUTION OF COMPUTING IN GLAXO PHARMACEUTICALS

In 1978 three of Glaxo's UK pharmaceutical companies; Glaxo Laboratories Ltd, Allen & Hanbury's and Evans Medical Ltd, were merged to form a new company; Glaxo Operations UK Ltd. It was decided at that time that new computer systems should be developed to provide extensive on-line facilities and form an integral part of the company's operations.

The three companies at that time had their own computers; a variety of models from IBM, ICL, DEC, Redifon, Data General and CMC; and most of the data processing was carried out in batch mode. Each company had separate applications systems and its own development and operations personnel.

Concurrent with this company reorganisation, Glaxo had a small team investigating the potential use of mini-computers. Limited experience of on-line applications on the existing mainframe computers had established that, although they were excellent at batch processing, they would not be as cost effective as mini-computers in processing the on-line applications systems of the new company.

The first HP3000 system, a two megabyte Series 3, was installed at Greenford, Middlesex in October 1978 to commence systems development.

During the period since 1978, the Company's data processing operations have been rationalised and a considerable transformation has taken place. The three separate company's centralised DP operations have been replaced by a single integrated network of HP3000's supporting a distributed Company database.

The Company's operations have also been streamlined in other areas. Central planning of production has resulted in better equipment and manpower utilisation; resulting in better service to customers and reduced inventory levels. Improved up-to-date information has enabled the company to increase its competitiveness in the marketplace and to improve its profits in a depressed UK economic environment.

The distribution of computers at a number of geographically separate offices and factories has brought in-house computer power to many users and the inter-site and inter-machine communications links have made the whole Company database available at any node of the network.

The conceptual single Company database is distributed across the network in more than 250 Image databases and though some data duplication is necessary this is kept to the minimum required to sustain good response time from applications programs. The lack of a distributed database management system has not proved a problem and the facilities of the MPE file system, DS/3000 and Image combine well to ease the problems of centralised database administration.

Glaxo Pharmaceuticals Ltd and several other companies were formed during the last year from Glaxo Operations UK Ltd. One of these companies, Glaxochem Ltd, became responsible for the bulk manufacture of pharmaceutical compounds and another, Glaxo Export Ltd, was formed to handle exports to overseas markets.

3. THE HP3000 NETWORK

The network is basically a dual star configuration with the central installation providing applications development, network and database management for the whole network, each installation of necessity having its own operations staff.

Computer terminal users are located at a number of geographically separate factory and office sites in the UK. The Company's headquarters are at Greenford, Middlesex; about 12 miles west of London, and there are manufacturing plants at Barnard Castle (County Durham), Speke (near Liverpool) and Ware (Hertfordshire). Each of these factories has a computer installation and there are also computers at the Glaxochem factories at Ulverston (Cumbria) and at Montrose (Angus). Other offices and factories served include those at Annan (Dumfriesshire), Cambois (Northumberland), Northolt, Ealing, Harefield, Beaconsfield and Holborn, London. There are also links to the Glaxo Export installation at Islington, London and to HP's offices at Pinewood.

The network today consists of 17 HP3000 computers. Three of these are used exclusively for applications development and a fourth for user training and database reorganisation work. At Greenford the four production computers are used for (1) Sales and Finished Stock Control, (2) Production planning, (3) Accounting, Marketing and Personnel systems and (4) Representatives Support and Clinical Trials. At the factory installations the Series 64 handles Production Control, Quality Assurance, Purchasing, Stores records etc and the Series 44 computer handles Accounting and Personnel applications. All machines provide on-line information and electronic office facilities such as HPMAIL and HPWORD.

The inter-site communications links between computers operate at 9,600 bits per second over leased lines using Racal modems. On-site the computer-to-computer links are hardwired and operate at 56,000 bits per second. Remote terminals are linked to the computers on a point-to-point basis using leased lines, Racal modems and statistical multiplexors. Further dial-up lines with modems provide stand-by for the leased lines and connection to Hewlett-Packard.

The on-line disc storage capacity is 11,963 megabytes. This includes 52 HP7925's, 14 HP7933's and 1 HP7912 disc drive. HP7976A tape decks and HP2608A line printers are used at each installation.

The network is operational 24 hours per day from Monday to Friday and from 8am and 8pm at weekends. During the week an on-line service is provided from 7:30 am until 6pm. At 6pm full dumps are carried out on all machines and database logging is restarted on new log files. This operation takes about two hours. A batch processing environment is operated for the remainder of the overnight period and at weekends.

There are at present 1066 computer users with access to 602 computer terminals. More than 1100 on-line and 950 batch programs have been developed. Extensive use is made of V/3000 for on-line work and all programs use IMAGE/3000 to process the company's data. This is made up of 3551 different data items forming over 7,000,000 entries in 1459 datasets of 251 Image databases.

Although data processing is carried out at a number of geographically distributed centres, the network's operation is nevertheless highly integrated by the use of applications programs that access and update both local and remote databases on-line.

4. THE USER INTERFACE AND OTHER CONCERNS

4.1 MPE COMMANDS

In a network environment with a distributed database, a terminal user would need to use the MPE commands; DSLINE, REMOTE, FILE and RUN to set up and execute applications programs. Some or all of the following information must be know to the user when entering these commands:

(a) Which computer his or her terminal is attached to.
(b) Which computer the required programs and form files are on,
(c) What Group(s) and Account(s) they are held in,
(d) What databases and other MPE files the programs use,
(e) Whether a program uses routines from a group or account SL,
(f) Which computer each database or data file is on,
(g) What Group(s) and Account(s) these are held in,
(h) What database password(s) to use,
(i) What DS line name(s) to use in linking to other computer(s),
(j) What remote log-on's should be used.

4.2 UDC's

UDC's (User Defined Commands) can be set up to issue these commands and some can be executed automatically at log-on time. Attempts were made at using UDC's but it proved to be impossible to achieve the very simple user interface that we wanted. In the early days we made greater use of UDC files but these were large, time consuming to maintain and extended log-on time for the user. When DS lines, remote sessions and remote access to data was catered for by the UDC's, MPE operating system table limits were encountered also.

4.3 SECURITY

The security provisions of the MPE accounting structure enable users to be given capabilities, and controls to be applied to file access. Whilst these provisions are probably excellent for stand-alone computers, with users that have their own programs, databases and other files, in a multi-user network environment where on-line access to shared data is catered for from local and remote computers, MPE security alone is inadequate.

IMAGE database management system can be used to provide additional data security and at Glaxo Pharmaceuticals 95% of all data is held in databases. Database logging is also used to ensure that data can be recovered when necessary.

An important consideration with a distributed database is how new applications programs can be tested as there is obviously a risk to the security and integrity of the Company's data if testing is carried out using live data. The majority of our systems development work is done at our central installation on separate development computers and these have separate smaller versions of the live databases containing test data.

How it is possible to develop and test applications in this single machine environment that will also work with the distributed live databases on the network with no code changes will become clearer shortly.

4.4 NETWORK MANAGEMENT

Important objectives of network management are to make the best use of available computers and be able to integrate new computers into the network when required. The network manager needs therefore the flexibility to be able to move users, software and/or databases from one computer to another in order to maintain a balanced workload on each machine across the network. How this can be done ralatively easily without changes being made to applications programs will also be clarified.

5. GLAXO PHARMACEUTICALS USER/NETWORK INTERFACE

At Glaxo Pharmaceuticals, many computer users always use the same computer terminal but a sizeable number, including directors and senior managers, need also to use terminals in other departments and at other offices and factory sites. To make this possible and to address the difficulties outlined previously we have developed and implemented an application harness which we have called MENUDRIVER. This is linked to MPE with a log-on UDC, and with an appropriate organisation of the MPE accounting structure provides a solution to the difficulties previously outlined. At the same time it provides a simple yet secure interface to the network for computer terminal users.

5.1 MPE ACCOUNTING STRUCTURE ORGANISATION

Users are given a log-on that utilises their initials as a USERNAME and a single letter as an ACCOUNTNAME. This letter is the initial letter of the town where the computer installation is located that the terminal they normally use is attached to. For example, a user at head office would use the "G" (for Greenford) account and one at Barnard Castle the "B" account etc.

Each computer also has ACCOUNT's for the initial letter of all other installations. The main account of each installation (G at Greenford) has GROUPS to hold the databases and applications software of the installation. The other accounts have log-on UDC's attached which establish a DS link to the computer installation indicated by the ACCOUNT name and a session on the remote computer. For example, a user signing on to the S ACCOUNT at Greenford will be connected through to a computer at the Speke installation. Apart from the slowing down effect of the communications line, the user will then work as if he had logged on directly at the remote computer installation.

The main account at each installation has a log-on UDC that launches the MENUDRIVER applications harness. Users remain within the control of this program until their data processing work is completed and on exit from the program are loged-off from the computer. Users are therefore unable to enter commands directly to MPE on their terminals. To cater for user, database and software administration there are of course a number of special log-on's but the use of these is restricted to a limited number of network and database administration staff.

5.2 THE MENUDRIVER APPLICATION HARNESS

The application harness consists of a program PMENU and five Segmented Library (SL) routines which are called by applications programs. A log-on to one of the installation main accounts will automatically cause the session concerned to execute PMENU.

5.2.1 THE STANDARD SELECTION SCREEN

The PMENU program presents the user with a standard selection screen onto which he or she may enter the name of an application program to be executed. PMENU then checks to see that the user is authorised to run the selected program by accessing a database called MENU. This database structure is held on each computer and lists for users at that installation the programs that they have been authorised to use. If a user selects a program that is not on his menu an error message is displayed and the selection is rejected.

If the user cannot remember the name of a program, HELP can be entered on the selection screen and PMENU will display a list of the programs that he is authorised to run together with a brief description.

5.2.2 THE MENU DATABASE

The MENU database also provides PMENU with:
(a) A list of databases used by the selected program,
(b) The DBOPEN mode for each database used by the program,
(c) Whether the access to each database is optional,
(d) The users·password for the DBOPEN mode for each database,
(e) The computer on which each database is stored,
(f) The DS line routing to the other computer(s) involved.

5.2.3 PROGRAM ENVIRONMENT INITIALISATION

When the selected program has optional access to databases, PMENU will list the optional databases and request the user to indicate whether each database is required for this run of the program.

If a database is not on the computer running PMENU, but is on one that is directly connected to it, a DS line will be opened to that computer and a remote session established thereon.

If a database is on a computer that can only be accessed via another, a route via the intermediate computer will be set up. A DS line to the intermediate computer will be opened and a session established thereon. Using program-to-program communication, a slave process will be launched on the intermediate computer. This will open a DS line to the computer where the database is located and establish a session thereon.

File equations will then be issued identifying the MPE group and account where all databases and other files are located. If remote computers are involved these equations will define the DS device involved. If an intermediate remote computer is involved the slave process running on that computer will issue a further file equation defining the on-going DS device to the destination computer.

PMENU will then launch the applications program selected as a son process.

5.2.4 THE SEGMENTED LIBRARY ROUTINES

The five SL routines that are contained in the application harness are designed to enable the user to thread his way through a series of programs without having to interact with MPE and to provide database passwords to programs. The routines are essentially menu-handling and process-handling routines that operate on a shared extra data segment. The extra data segment is used to communicate information from one process to another within the same process tree.

The majority of application program selections entered by the user via standard selection screen simply cause the selected program to be loaded and executed. MENUDRIVER provides for the transfer of control between one program and another.

Some applications programs may, however, have a number of processing options. MENUDRIVER also provides for the transfer of control from a program to itself, eliminating the process handling overhead in this event.

One single menu can therefore be maintained for a user and the selections therein can refer to a mixture of programs and also processing options within the same program.

5.2.4.1 SL Routine SETMENU

This is an initialisation routine. It is called at the beginning of each applications program and at the beginning of PMENU.

If the shared extra data segment used by MENUDRIVER does not already exist, then this routine will build it using infomation obtained from the MENU database.

The extra data segment is used to hold a nucleus of information which reduces the need to keep referring back to the MENU database during subsequent SL calls

On each call to SETMENU, the current contents of the extra data segment are copied into the calling program's working store and the previous son process if any is "killed".

5.2.4.2 SL Routine GETCHOICE

This routine provides the menu-handling interface between the application program and the user.

The routine first tests the "selection" field in the extra data segment to determine if the selection is currently in force and not yet actioned. If so, the routine exits to the calling program without taking any action. If the selection field contains blanks, which is a signal from the calling program that the user's last selection has now been fully executed, the routine displays the standard selection screen.

The next program selection may then be entered on the selection screen. Alternatively, "HELP" may be entered to obtain a described list of the programs that the user is authorised to run.

When the user enters an authorised program name it is placed in the "selection" field of the extra data segment and the routine exits to the calling program

Where an applications program has a number of processing options, the calling program must use the "selection" field to determine which of the options has been chosen.

The extra data segment also contains a field called "driver reply" which is used by GETCHOICE to signal to the calling program that the required selection resides in another program. To indicate this a value of "LINK" is placed in this field and the applications program must then call LINKOUT to transfer control to the selected program.

In each session the first call to GETCHOICE occurs in the PMENU program.

5.2.4.3 SL routine PUTCHOICE

Normally when the selected processing is completed, the applications program would call GETCHOICE to ask the user for the next selection to be made on the standard selection screen it displays. In some cases this would result in a dialogue that is considered wasteful in terms of generated message pairs.

PUTCHOICE can be used to eliminate the standard selection screen. Application programs using this must provide for entering a "next selection" field on one of the applications screens. The value entered is presented to PUTCHOICE for verification and the driver reply will tell the program if the selection is valid and if it involves a transfer of control to a separate program.

5.2.4.4 SL routine LINKOUT

The purpose of this routine is to transfer control between one process and another. The effect of the routine is to sustain a simple process tree containing a father and, intermittantly , a single current son process.

When the routine is called from PMENU it launches the program indicated in the "selection" field of the extra data segment as a son process and suspends itself.

When the routine is called from a son process, it returns control to PMENU.

5.2.4.5 SL routine BASEPASS

This routine eliminates the need to either build-in database passwords to programs or to ask for them from the user at run time.

Database passwords are held on the MENU database and obtained at run time with this routine.

Users are not aware of database passwords and do not need to know them. The Database Administrator can therefore control access to data by users by setting up database passwords with appropriate user access capabilities.

This allows the Database Administrator overall control of access to the Company's database; stopping an applications program, when a user is inadvertantly authorised to run the program but is not authorised to access the data that the program uses.

6. SUMMARY OF BENEFITS

6.1 The simplest possible users' interface to the network.

6.2 A single unique log-on for each user.

6.3 User log-on at any terminal (excluding development terminals).

6.4 Usage control of applications programs.

6.5 Access control to databases by Database Administration.

6.6 A Standard application selection screen with with HELP facility.

6.7 Elimination of computer, group and account qualification by applications programs, in UDC's or by users.

6.8 Ability to move databases without program code changes.

6.9 Ability to change communications links without program code changes.

6.10 Ability to develop and test applications on a single machine and know they will run without code changes on the network computers.

THE COMPUTER USER/NETWORK INTERFACE AT GLAXO PHARMACEUTICALS LTD

30,000 EMPLOYEES WORLDWIDE

MARKET VALUE: US$ 5,500 MILLION

ANNUAL SALES: US$ 1,400 MILLION

ONLY BRITISH GROUP IN
THE WORLD'S "TOP 20"
PHARMACEUTICAL COMPANIES

SUBSIDIARIES IN
OVER 50 COUNTRIES

INTRODUCTION

EVOLUTION OF COMPUTING
IN GLAXO PHARMACEUTICALS

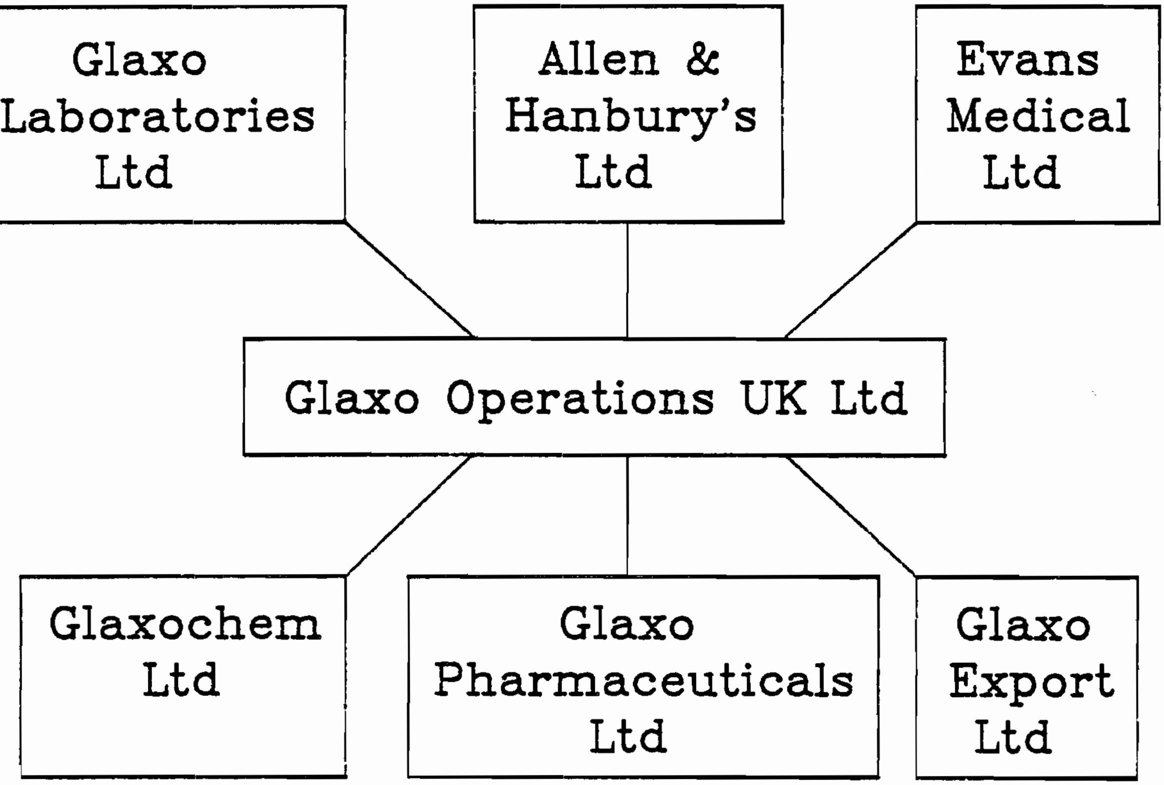
THE HP3000 NETWORK

THE USER INTERFACE
AND OTHER CONCERNS

GLAXO PHARMACEUTICALS
USER/NETWORK INTERFACE

SUMMARY OF BENEFITS

GLAXO UK COMPANY CHANGES IN RECENT YEARS



Glaxo Laboratories Ltd

Allen & Hanbury's Ltd

Evans Medical Ltd

Glaxo Operations UK Ltd

Glaxochem Ltd

Glaxo Pharmaceuticals Ltd

Glaxo Export Ltd

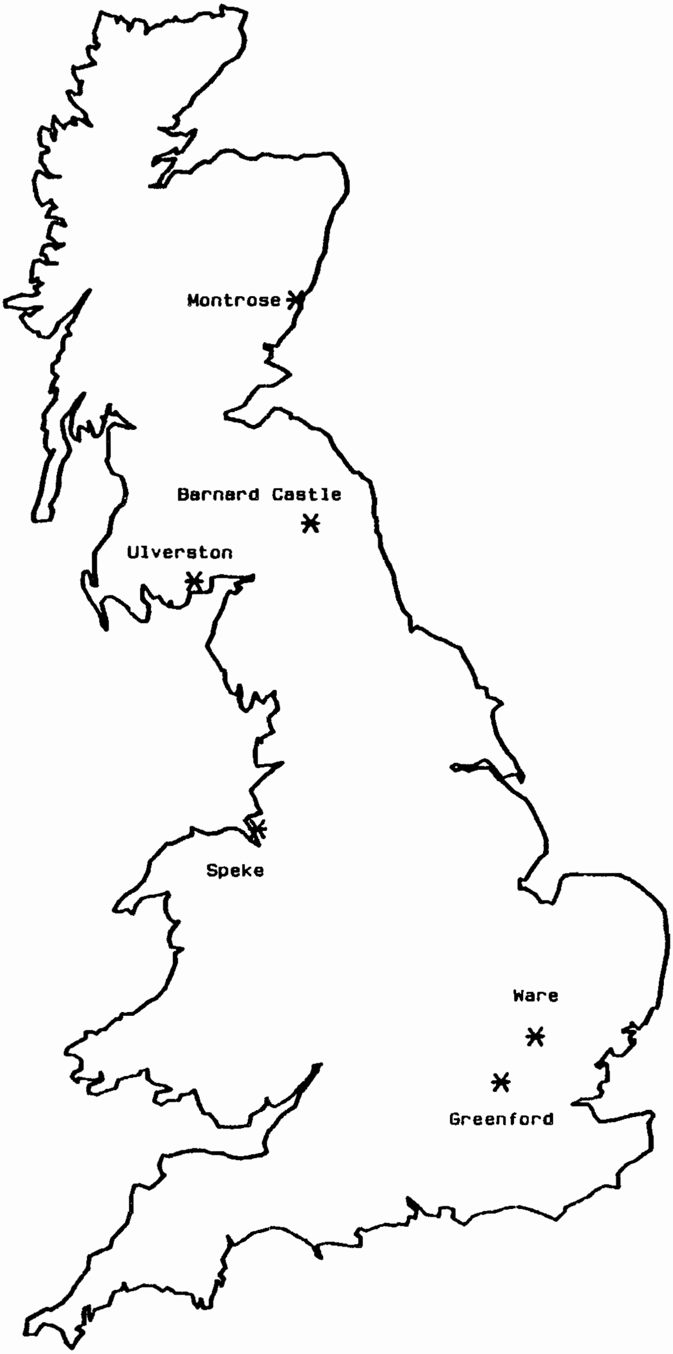
1978

GLAXO LABORATORIES LTD
ALLEN AND HANBURY'S LTD
EVANS MEDICAL LTD

BATCH PROCESSING ON ICL MAINFRAMES

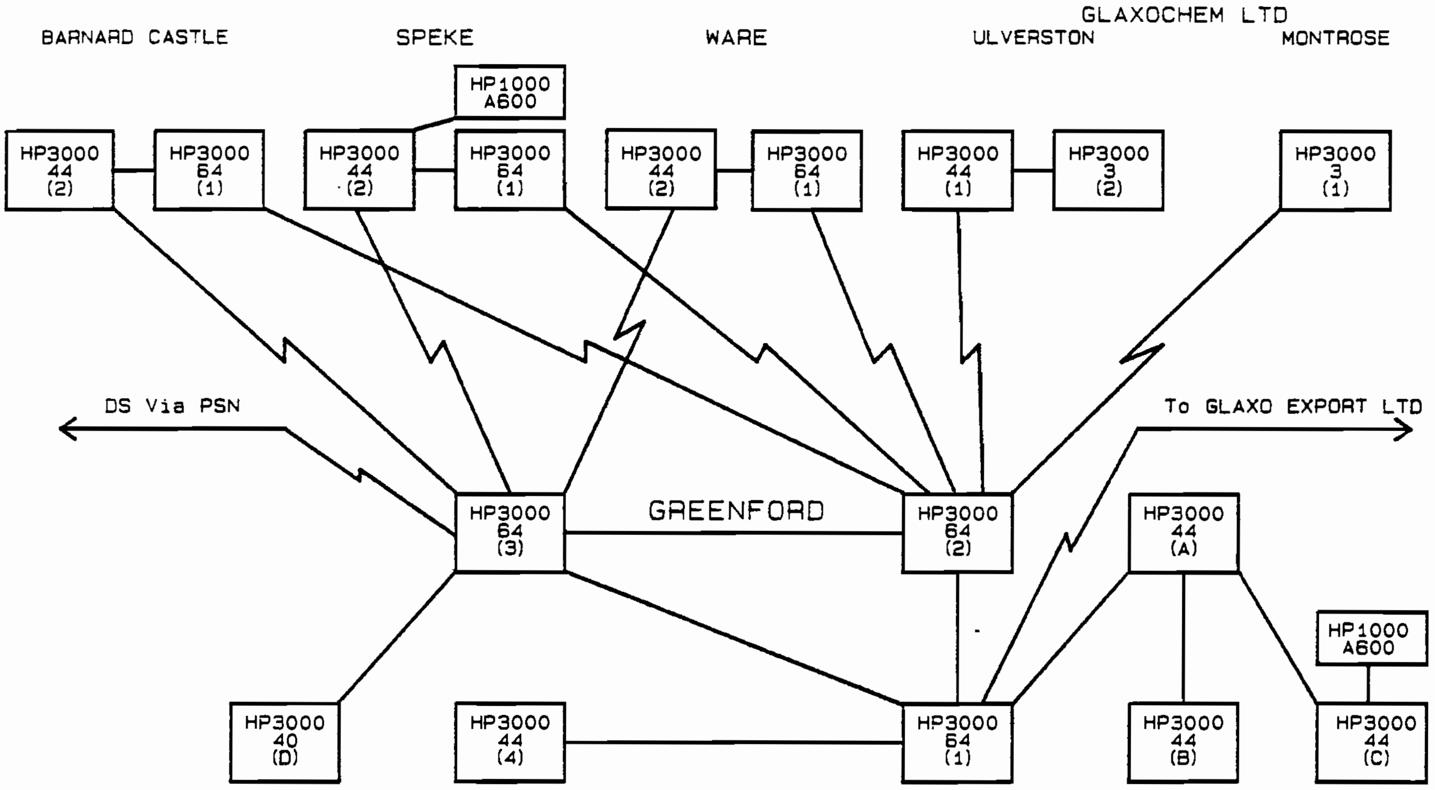
ALSO IBM, DEC, DATA GENERAL, CMC
& REDIFON MINI-COMPUTERS

SEPARATE DATA PROCESSING OPERATIONS


Montrose
Barnard Castle
Ulverston
Speke
Ware
Greenford

GLAXO PHARMACEUTICALS LIMITED

COMPUTER NETWORK - AUGUST 1983


BARNARD CASTLE
SPEKE
WARE
GLAXOCHEM LTD
ULVERSTON
MONTROSE
HP3000 44 (2)
HP3000 64 (1)
HP3000 44 (2)
HP1000 A600
HP3000 64 (1)
HP3000 44 (2)
HP3000 64 (1)
HP3000 44 (1)
HP3000 3 (2)
HP3000 3 (1)
DS Via PSN
To GLAXO EXPORT LTD
GREENFORD
HP3000 64 (3)
HP3000 64 (2)
HP3000 44 (A)
HP1000 A600
HP3000 40 (D)
HP3000 44 (4)
HP3000 64 (1)
HP3000 44 (B)
HP3000 44 (C)

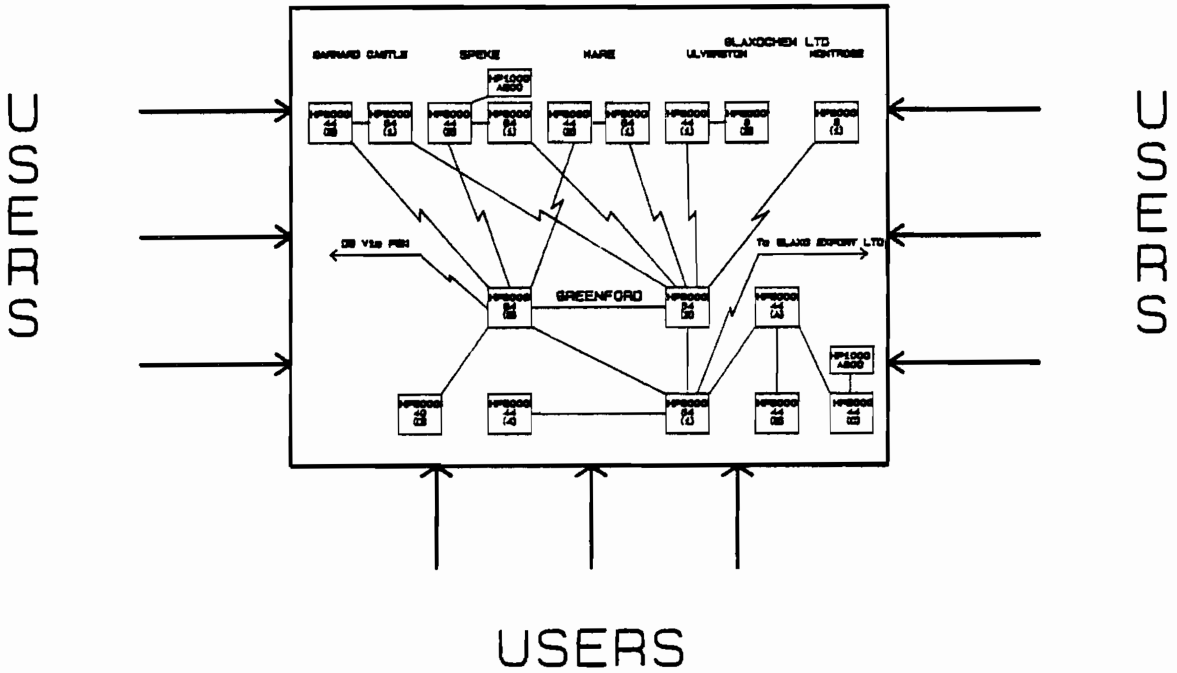
THE USER INTERFACE

MPE COMMANDS

UDC's

SECURITY

THE USER/NETWORK INTERFACE


GLAXOCHEM LTD
BARNARD CASTLE
SPEKE
WARE
ULVERSTON
MONTROSE
HP1000 A900
HP3000
DS Via PSN
To GLAXO EXPORT LTD
GREENFORD
USERS
USERS
USERS

GLAXO PHARMACEUTICALS
USER/NETWORK
INTERFACE

MPE ACCOUNTING STRUCTURE

LOG-ON UDC's

APPLICATIONS HARNESS

NETWORK MANAGEMENT

"MENUDRIVER"

"PMENU" PROGRAM

"MENU" DATABASE

5 SL ROUTINES

USER LOG-ON EXAMPLES

"HELLO DWC.G"

"HELLO DS.B"

"MENUDRIVER" FEATURES

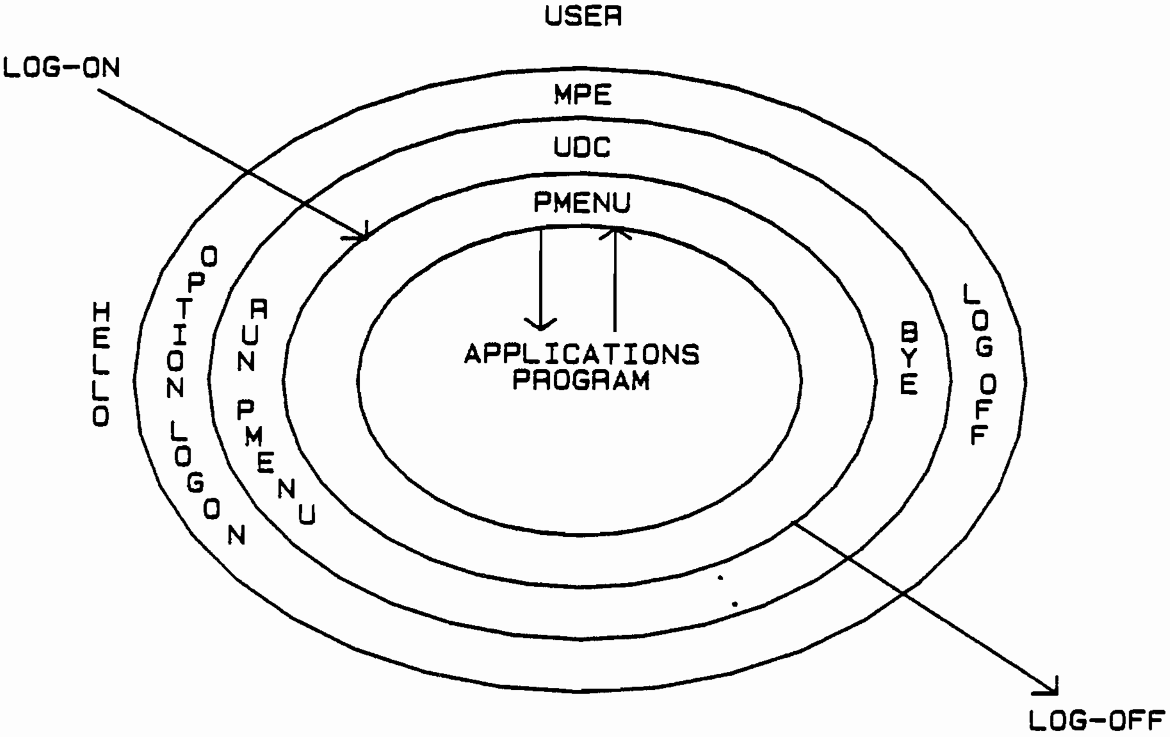
STANDARD SELECTION SCREEN

PROGRAM USAGE CONTROL

DATA ACCESS CONTROL

NETWORK DATABASE ACCESS

USER INTERFACE


USER
LOG-ON
MPE
UDC
PMENU
HELLO
OPTION LOGON
RUN PMENU
APPLICATIONS PROGRAM
BYE
LOG OFF
LOG-OFF

WITH MENUDRIVER

"MENUDRIVER" SL ROUTINES

SETMENU

GETCHOICE

PUTCHOICE

LINKOUT

BASEPASS

A COMMUNICATIONS PACKAGE FOR INTERNETWORKING ON THE HP3000

Reinhold Leitner
VOEST-ALPINE
Linz/Austria

ABSTRACT

We have developed a communications package which allows program-to-program communication with different computers with various operating systems using identical subroutine calls.

INTRODUCTION

In our company there are different types of computers from various manufactures. Each computer has different communication capabilities providing different user interfaces and services.

For example:

HP3000 using DS/3000
PDP-11 and VAX using DECnet
IBM using SNA etc.

It was our goal to develop a communications package with following features:

- an interface for program-to-program communication with identical subroutine calls on various operating systems (e.g HP3000/MPE, DEC/RSX etc.)
- full routing capability to all computers connected to this network
- low cost by using existing communications hardware and software (eg. HP's DS/3000, DEC's DECnet etc.)
- no modification of operating systems using only supported system calls (intrinsics).

How can we achieve this ? Let us take a look at the structure of a layered communications software.

ISO Reference Model of Open Systems Interconnection (OSI):

OSI defines seven layers:

layer		some functions
7	Application	filetransfer,virtual terminal, remote job entry etc.
6	Presentation	code conversion
5	Session	Connect, Disconnect a logical link
4	Transport	end-end error control and flow control
3	Network	routing and switching
2	Data Link	link error detection and recovery, link flow control
1	Physical	mechanical, electrical characteristics

We did not like to invent the wheel another time. We would like to use existing communications hardware, communications software and protocol emulators.
But we need a clamp to tie all the software modules together to form a network.

Therefore we defined a simple protocol (so called VOEST-ALPINE Standard Protocol (VAS)) which resides (from the users point of view) at level 5 (session layer).
(It is a simplification to say that the VAS protocol is a level 5 protocol. The specific implementation of this protocol depends on the used lower level protocols, depends on the services provided by the lower level protocols. For example if the used lower level protocol does not provide flow control, VAS has to be extended to provide this function. Another extension is necessary to provide routing by special VAS-gateways and end-end error control.)

VAS implements following set of services (intrinsics):

INTERFACE TO THE VAS-PROTOCOL

We will give a brief description of the functions and parameters of the VAS-Subroutines:

Parameters for all Subroutines:

communication area (I for input to subroutine,
status array (O) O for output from subroutine)
linknumber (I,O)
mode (synchronous or asynchronous call) (I)

Subroutine-(Intrinsic-) name and parameters:

Connect:
(Establish a logical link (session))
- type of connect (connect to send or receive data) (I)
- local channel name (used to distinguish different functions within sender program) (I)
- remote node name (I)
- remote program name (I)
- remote channel name (used by remote program for special functions eg. filename) (I)
- remote account information (remote hello) (I)

Disconnect:
(Terminate a logical link)
- status code to acknowledge data, if data received (I)

Abort:
(Kill a logical link, may be called by master and slave at any time)
- error code (I)

Send Data Message:
- data buffer (I)
- lenght of data buffer (I)
- indicator if last or not last data message (I)

Send Status Message:
(Status to acknowledge incoming connects or received data)
- statuscode (I)

Receive Data Message:
- data buffer (O)
- lenght of data buffer (I)
- (lenght of received data message within status array (O))

Receive Status Message:
(Status will be received within status arrary)

Wait:
(Wait for incoming calls)

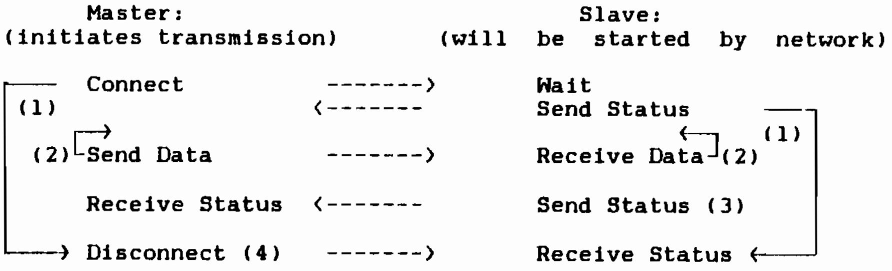
Get Link Information:
- local node name (O)
- local identification (account info) (O)
- local channel name (O)
- remote node name (O)
- remote identification (account info) (O)
- remote channel name (O)

Subroutines to open and close the network

Subroutine to request further transmission of data on an existing link

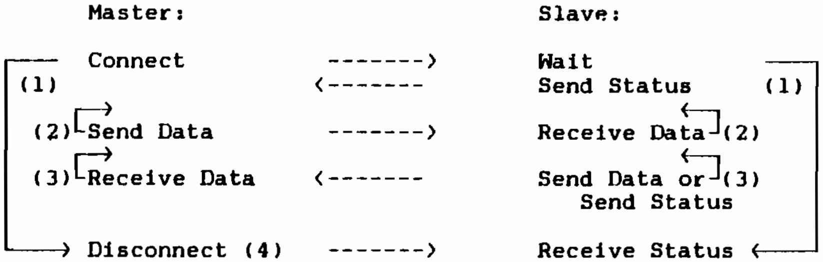
Sample sequence of calling this subroutines:

- to send data


Master:
(initiates transmission)
Slave:
(will be started by network)
Connect
-------->
Wait
(1)
<--------
Send Status
(1)
(2)
Send Data
-------->
Receive Data
(2)
Receive Status
<--------
Send Status (3)
Disconnect (4)
-------->
Receive Status

(1) If the slave program responds with a 'negative' status message, the master program has to disconnect the line (or to try again to send data or request data).
(2) Send (receive) data messages until master sends last message.
(3) Slave sends status message to master after processing the received messages.
(4) Master disconnects the line.

- to request data


Master:
Slave:
Connect
-------->
Wait
(1)
<--------
Send Status
(1)
(2)
Send Data
-------->
Receive Data
(2)
(3)
Receive Data
<--------
Send Data or
(3)
Send Status
Disconnect (4)
-------->
Receive Status

(1) If the slave program responds with a 'negative' status message, the master program has to disconnect the line (or to try again to send data or request data).
(2) Send (receive) data messages until master sends last message. This messages can be used as 'keys' to retrieve data.
(3) Slave sends 'negative' status message or sends data messages.
(4) Master sends status message to slave after processing the received messages (if any messages received) and disconnects the link.

Extension (not shown in the examples above):

You need not disconnect the link if you have sent or received a 'complete' set of messages and exchanged the status message. You can request further transmission of data on an existing link without issuing a new connect.

Summary of functions:

Using the VAS-protocol you can send data to remote programs and request data from remote programs.
When data is received the data should be processed first and then the status message should be sent. So you can achieve 'end-end-control' or write 'checkpoints' and the sender has the guarantee that the data is processed properly (stored to disc etc.).
Multiple logical links are supported simultaneously within a program and multiple active programs within a computer.

SPECIFIC IMPLEMENTATION ON HP3000/MPE

We have to consider two different cases:

- communication within a network of HP3000's
- communication with different computers

Communication within a network of HP3000's (and HP1000's):

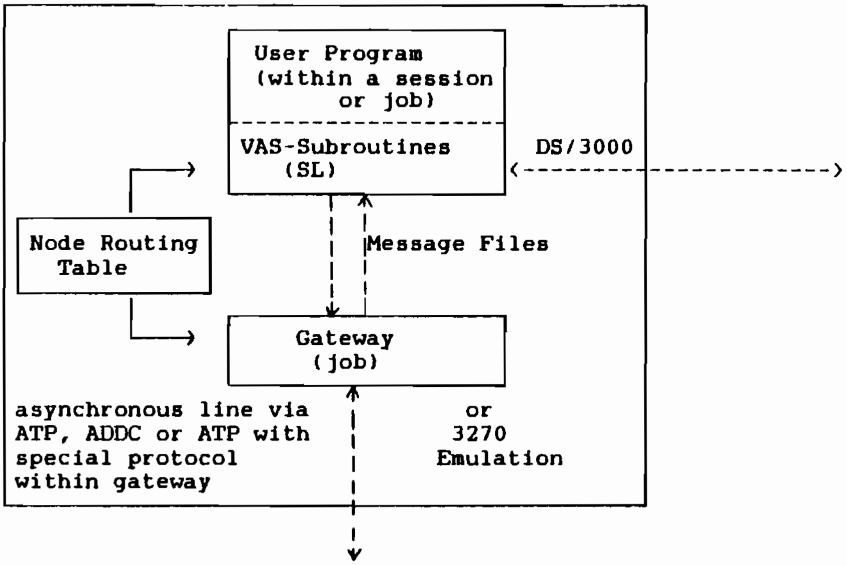
We can use HP's proprietary network DS/3000 using PTOP (Program to Program Communication). A VAS-Connect will be converted to DSLINE, REMOTE HELLO and POPEN etc.
The only problem is: There is no routing capability within DS/3000. This problem is solved using routing tables and routing jobs.

Communication with different computers:

We could use existing emulators with a task interface (eg. IMF/3000 which emulates IBM's 3270 slave or MTS/3000 which emulates IBM's 3270 master).
The existing implementation of X.25 on HP3000 is no solution of our problem, because X.25 is integrated within DS/3000. Therefore DS/3000 has to run at the remote computer also.

But there is another way to communicate with a HP3000 without writing drivers and/or using special hardware:
Use an asynchronous terminal port (V.24)!
This solution is cheap, but we have to implement a protocol to guarantee error-free transmission over this asynchronous line.

Structure of the software within HP3000:


User Program
(within a session or job)
VAS-Subroutines
(SL)
DS/3000
Node Routing Table
Message Files
Gateway
(job)
asynchronous line via ATP, ADDC or ATP with special protocol within gateway
or
3270 Emulation

The VAS-Subroutines:

The subroutines are written in FORTRAN and reside within a segmented library. When a user program calls connect the node routing table is looked up to determine which DS-line or gateway must be used to reach the remote node.

The Gateway-Program:

The gateway program is written in FORTRAN. The concept of a finite state machine is used. Within the gateway there is a clear interface between the VAS-protocol and the lower level protocols. So the lower protocols can be replaced easily.
Each protocol level is implemented using state tables.

For example:

```
STATE 32
        IF SND          THEN REC                  STATE=32
        IF <REC,STP>    THEN <WLB,<TIME,T2>>      STATE=33
        IF <REC,SX>     THEN <WLB,<TIME,T1>>      STATE=6
        ELSE FAIL2
```

This part of the state table has following meaning:

> If the VAS-protocol is within state 32 and sending a message to a user program via a message file has completed (SND), then issue FREAD (REC) to another message file to receive data from the user program. The new state is state 32.

- If the VAS-protocol is within state 32 and the received VAS-protocol message via messagefile is 'status positive' (<REC, STP>), then write this message to the line (to the protocol level below) which connects to the remote gateway and set the timer to value T2. The new state is state 33.

- If ....

- If a condition occurs which is not mentioned within the table entry for the current state, the subroutine FAIL2 is called for errorprocessing (e.g. termination of the logical link or retransmission of a message).

To implement this concept, a parser has been written which translates these state tables and generates data structures. These translated tables are interpreted at run-time.

Implementation using an asynchronous line:

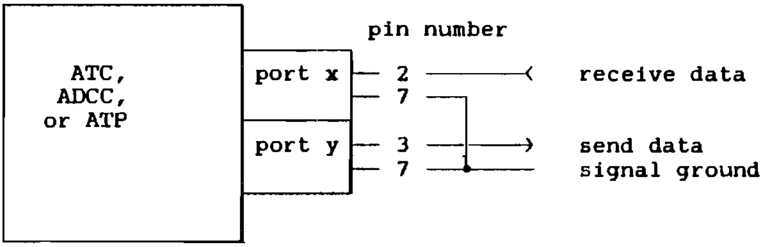
We can use any HP3000 terminal line on an ATC, ADCC or ATP. To guarantee error-free transmission we had to define a protocol for this asynchronous line which must provide following services to the layer above (VAS-protocol):

- communication must be full duplex
  (send or receive at the same time)
- error-free transmission
  (linkerror detection and recovery)
- link flow control

How can this functions be achieved on a HP3000 computer ?

- full duplex:

  When a read is pending you must have the possibility to write to the terminal. This is not possible on the HP3000 with one line. So we are using two lines with proper V.24 connectors:



  We do not want to loose any incoming data. So we opened the line for read twice. In spite of having two pending reads at any time, some characters are lost, if two messages are transmitted to the HP3000 one after the other without delay. So the remote gateway sends some NUL characters in front of each messages to give time to MPE to switch to the next read. (This leading NUL characters are ignored by the MPE terminal driver).

We don't want any ENQ/ACK handshake etc. So we use termtype 18. And we have to use nowait-I/O to accomplish this task.

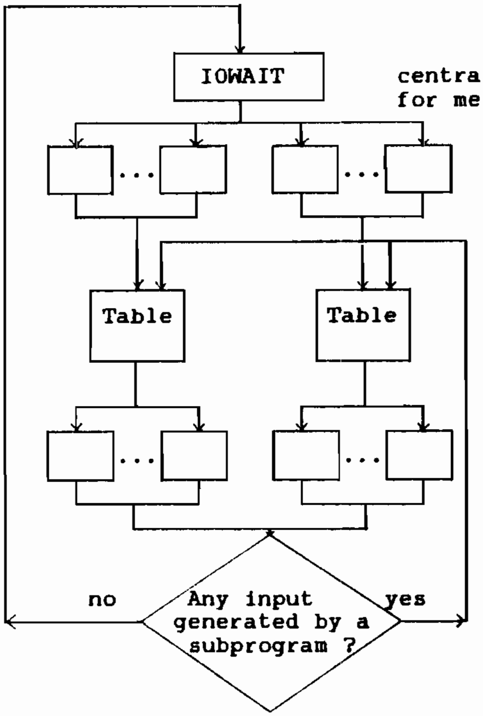
Summary:

- open one line for write-only, CCTL, ASCII, new, nowait I/O
- open another line for read/write (write needs no operator reply), ASCII, new, nowait I/O, share
- allocate this lines setting termtype 18 using FCONTROL
- set unedited mode with special end-of-record character using FCONTROL
- set echo off, disable system break, disable subsystem break, set parity and enable parity checking using FCONTROL

- error-free transmission and link flow control

For link error detection we currently use parity and checksum. (Warning: There is no possibility to check parity on HP3000/III ATC when using unedited mode !!!).
Retransmission is accomplished using ACK/NAK handshake and timers. (Timer service is accomplished using timeouts on a message file.)

Simplified flowchart for gateway:



central IOWAIT call
for message-files and terminal

IOWAIT completion
preprocessing (generating
input for table)
depending on filenumber.

Table interpreter calls
various subroutines.
These subroutines can
generate input for all
protocol levels.

Postprocessing
depending on filenumber.

Summary of functions on HP3000 which are implemented by now:

- VAS-subroutines which form the interface to the network
- Connection via gateway supporting an asynchronous line to another HP3000 (this gateway is symmetrical!) or to any computer where a corresponding gateway is implemented (e.g. DEC PDP-11 with operating system RSX-11M)
- Connection to remote HP3000 using DS/3000 and HP1000 using DS3000 TO DS/1000

In all cases route through to non-adjacent computers is supported.

IMPLEMENTATION OF GATEWAYS SUPPORTING ASYNCHRONOUS LINE ON NON HP COMPUTERS

Gateway on DEC PDP-11 (with operating system RSX-11M) which converts logical links to DECnet and allows access to all computers connected to this network
Access via DECnet to SIEMENS and IBM computers is available.

FUTURE PLANS

The network is steadily expanded. Future plans concern with the integration of WANG/OIS, UNIX, IBM/SNA etc.

SYNCHRONOUS COMMUNICATION ON THE HP 3000

N. M. DEMOS

SYNCHRONOUS COMMUNICATION ON THE HP 3000

N. M. DEMOS
DEMOS COMPUTER SYSTEMS, INCORPORATED

The interface between a terminal and a computer has remained a mystery to many. This is only as it should be. The typical user should not have to know how data gets to and from his terminal. It is sufficient that it does. Others, such as those responsible for configuring and ordering hardware, need only a very basic knowledge. Sometimes a more thorough knowledge permits the implementation of applications not otherwise possible. We will try to enlighten our reader so that he will have enough background to be able to know how to proceed further if he is so inclined.

Most terminals are attached to the HP3000 by a direct connection ("hardwire"). They use only three conductors - one conductor to transmit data, another to receive data and the third as a common signal ground. The data is sent serially bit by bit. That is, the character is sent by the 3000 one bit at a time, using the American Standard Code for Interchange of Information (ASCII). ASCII uses seven binary bits to represent all the letters of the alphabet, both upper and lower case, numbers, special characters and other (normally unseen) control characters. The control characters indicate functions such as carriage return, line feed and end of message. In addition to the seven bits, a bit is normally reserved for parity, and a start and stop bit are required to indicate to the receiving device that a character is beginning and ending. At low speeds (110 characters a second), two stop bits are used. This is the "asynchronous" mode of communication.

In order to communicate over telephone lines, additional conductors (typically three or four) are required to control the modem (the device that converts the computer digital signal to and from the telephone analog signal). The voltage and way that these are connected (interfaced) from the 3000 to the modem, from the modem to the remote terminal or directly from the computer to the terminal conform to the Electronic Industries Association (EIA) standard RS232C. In this way terminal and modem manufacturers can design and sell equipment for any computer that, like the 3000, adheres to this widely used standard. However, all the demands of data processing cannot be met by this simple approach. It was designed for speeds approximating a Teletype (maximum thirty characters a second). With the advent of higher speed devices, particularly the CRT display, the asynchronous mode and the RS232C standard caused severe problems. The RS232C standard will not support long distance communication at high speed (one reason for the introduction on the Series 64 of the RS422 standard for terminals attached directly).

For high speed transmission over telephone lines, the asynchronous mode is not effective. In the first place this method imposes a limit of 120 characters per second because each character is individually timed. This is not an absolute limit, as new technonogies become available, but there is an inherent speed constraint. Secondly there was little requirement at the low speeds originally used to implement checking to insure that a message was correctly received and request a retransmission if there was an error. Also the requirement for a start and stop bit for each character reduced the effective throughput by twenty five percent.

In order to attain increased speeds the synchronous method of transmission was implemented. This method requires that the devices at both ends stay synchronized for an entire message (typically a line of data, but often much more). With synchronous transmission very high speeds are possible. For example, the HP 3000 supports speeds up to 7000 characters per second in synchronous transmission. As speeds increase, it is more important to automatically check for transmission errors and re-transmit when an error is detected. Because of this requirement and the need to maintain synchronization over an entire message, the hardware has to be more sophisticated and therefore more expensive than hardware used only for asynchronous transmission, but the payoff is in the higher speeds attainable.

The most common and one of the oldest methods ("protocols") used to take advantage of synchronous communications is the IBM Binary Synchronous Communications Protocol and its variations. HP uses this protocol for most of its communications products, both because it is well known and because HP wishes to support communications with IBM main frames. HP also supports a newer protocol (HDLC) patterned after IBM's Synchronous Data Link Communications (SDLC) protocol as well as SDLC itself. HDLC and SDLC make better use of more recent transmission technologies and are therefore more efficient. For a description of these protocols and others refer to the Guidebook to Data Communications (see the bibliography).

On the HP 3000, the INP (Interface Network Processor) is the current interface for synchronous communications. The INP has its own microprocessor and buffers to reduce overhead on the processor. It converts data to and from one synchronous line. On the Series III there is also the SSLC (Synchronous Single Line Controller) and the HSI. The former is not as flexible as the INP (speeds only to 1200 characters per second for example) and requires more process time (or "overhead") on the 3000. The HSI (High Speed Interface) is used only for a Series III communicating with another in the same building.

On the 2624 and 2626 terminals, there is a special feature that allows these terminals to operate synchronously. The 2645 also has a board that gives the same capability. HP has a new product (the Multipoint Cluster Controller - product number 2333A) that allows any standard asynchronous terminal (up to 16 per controller) to take advantage of synchronous transmission. Therefore to use synchronous transmission the hardware items required are terminals as described above and at least one INP on the 3000. To transmit over telephone lines, the appropriate modems (or digital units) and lines are required.

In order to implement various synchronous communication capabilities on the 3000, HP developed a subsystem called CS/3000. This consists of a series of intrinsics analagous to the file system intrinsics (FOPEN, FREAD, FWRITE, FCONTROL, etc.) that can be used to define the desired communications protocol, logical device, speed, etc. (COPEN) and receive and send data (CREAD and CWRITE). HP has an internal document that gives much information on how to use these intrinsics (CS/3000, External Reference Specifications). There is no HP product available called CS/3000. Instead the user obtains CS/3000 by purchasing any HP communications product. CS/3000 will be described later.

Multipoint Terminal Software (DSN/MTS) is the subsystem required to support HP synchronous terminals on the 3000. Except for some limited models of IBM's 3270, it only supports HP terminals. However, standard asynchronous terminals are supported through the Multipoint Cluster Controller mentioned above. MTS also supports hardwired terminals using HP's Data Link method (mostly for the 307X family of factory data collection terminals). Its major advantage for the non-factory user is its capability to support remote clusters of terminals with a single high speed line. It uses Binary Synchronous protocol (multipoint) except in special cases. It does support an asynchronous protocol for local application (but you still need a special product for your terminal to use it this way). It is also a method to operate terminals at 600 characters or 1200 characters per second (speeds not otherwise available) on the Series III if that is imperative.

The Remote Job Entry (DSN/RJE) subsystem is used to make the 3000 appear to an IBM mainframe like an IBM 2780 or 3780 Remote Job Entry work station. It can also be used to communicate between two 3000's but DSN/DS (see below) is usually a better choice for this purpose. Data may be sent from and received on almost all 3000 peripherals. The RJE concept involves batch processing, so this subsystem cannot be used for interactive communication.

RJE uses IBM Binary Synchronous protocol in what is called "Contention" mode. This means that each end of the line (the 3000 or the IBM mainframe) bids for the line. There is no polling as in MTS. Therefore only one 3000 can be on the line at a time communicating to one IBM mainframe. However, dial-up as well as leased telephone lines are supported so that connections to various host at different times can be made if required.

Multileaving Remote Job Entry (DSN/MRJE) is a more sophisticated version of RJE. MRJE emulates work stations that communicate with the IBM HASP, HASPII, ASP, JES2, and

JES3 Job Entry Systems. This package should be used in place of RJE when it is available on the host system and multiple users on the 3000 wish to submit jobs for processing. Because the 3000 emulates an intelligent system (computer) much more flexibility is available to control job submission, control and routing. It also cannot support interactive communication with an IBM mainframe because of the nature of the IBM system it emulates. MRJE uses an option of IBM Binary Synchronous protocol called "Conversational" mode. This mode makes the communication link more efficient in cases when there is data to be sent both ways. The normal receiving station can send data along with its protocol acknowledgements (this is where the term "Multileaving" comes from).

The Interactive Mainframe Facility (DSN/IMF) is the only other HP product that communicates with an IBM host. As its name implies, it is used to communicate interactively. It makes the 3000 look like an IBM 3270 type cluster controller. A terminal on the 3000 can look like an IBM 3270 type terminal (Pass Thru Mode) or a 3000 program can access the host with special intrinsics (Program Access Mode). Program Access Mode is a powerful tool. It is the only HP supported method to programatically communicate with an IBM mainframe interactively. IMF uses either Binary Synchronous or IBM SDLC to communicate with the IBM host. In both cases it supports multi-point, that is, the 3000 can be one of many stations on the same line.

Distributed Systems (DSN/DS) is the most powerful of the HP communication products. It is used for transferrring data between two 3000's and between a 3000 and another HP computer (1000, 250, etc.). It also supports compatible terminals on Public Data Networks (Telenet and Tymnet). With DS a user on one HP computer can log onto another HP computer, use any peripheral on the other computer, transfer files to and from the other computer, access a data base on the other computer and write programs that talk to each other directly over the communications link. The link may be either local ("hardwired") or over telephone equipment (remote). Remote access may be either over a leased line or by dial-up. Programs do not have to be changed to use most of the capabilities of DS. Extensions of the FILE Command and additional commands, e.g. REMOTE, provide these capabilities in a user friendly fashion. DS uses the Binary Synchronous protocol except for Public Data Network access, in which case the X.25 standard protocol is employed.

Originally all these communications products were to be programmed using the CS/3000 subsystem. Unfortunately, performance considerations forced HP to compromise. Both DS and IMF use modules that do not employ CS/3000 for interface to the INP.

What does the user do if none of the HP communications systems meets his needs? First of all he should make certain that he cannot use one of these systems, either by asking for changes at the host or by redesigning his application. For example, if he requires interactive access to an IBM mainframe, his first option is to ask that he be given access as a 3270. He can then implement his application through IMF. If this does not apply and he is a knowledgeable user he may be able to use the CS/3000 Intrinsics. This was done successfully in one case where it was required to communicate interactively with an IBM mainframe using the Binary Synchronous protocol in contention mode.

The application required an interactive response for credit card verification and data capture. IBM Binary Synchronous protocol using contention mode was the communications mode specified. This is the same method and protocol used by RJE, but RJE assumes communication to the host in a batch mode (remember RJE stands for Remote Job Entry). Although data can be entered through $STDIN, the host looks upon the input as a job submission. Also RJE could not be integrated into the rest of the application on the 3000, it is meant to be a stand-alone program. It may have been possible to have the IBM host communicate with the 3000 as a 3270 cluster controller, with the 3000 using the Interactive Mainframe Facility (DSN/IMF). IMF has intrinsics that can be used to make a program look like a 3270 terminal. However, IMF is an elaborate subsystem with a high CPU overhead. The IMF Program Access Mode (the way it accesses the host programatically) requires the programmer to use and format a dummy terminal screen. This requires considerable programming as well as adding to the overhead.

Performance was an important consideration because of the load on the system. Only because there was no feasible alternative was it decided to use CS/3000. This approach required considerable knowledge of communications and experimentation with some parameters of the CS intrinsics. It was successful in achieving its applications goals.

In order for the programmer to use CS/3000 he must first configure his INP (SSLC or HSI if he wishes to use one of those interfaces on a Series III). The configuration is fairly straight forward, and if he does make a mistake, most of the configuration parameters can be overriden with the CLINE command. Because we were going to use IBM Binary Synchronous protocol (Bisynch) we followed the specifications for RJE in the Systems Manager/Systems Supervisors manual. Our communications equipment consisted of an INP, its modem cable, a modem (201C equivalent) and a leased line. Therefore the TYPE in the configuration was 17 (INP) and the SUBTYPE was 1 (Synchronous, nonswitched line with modem). Switched means dial up, non-switched means leased line. Common sense applied to the rest of the parameters and they can be overriden by the CLINE command and/or the COPEN intrinsic anyway.

There is only one driver for the INP, it is called IOINP0. This is only the driver skeleton. The rest of the protocol is contained in a download file. For IBM Bisynch contention mode this is CSDBSC0, the default. It comes with DSN/DS and

DSN/RJE (and maybe other DSN products). If you are trying to use CS/3000 with other versions of Bisynch or with SDLC, beware. You will have to match your protocol and mode with one of the DSN products and purchase it to get the correct download file. A friendly, communications trained HP SE is invaluable at this point.

If you have any questions about the required communications equipment, you should refer to the HP Guidebook to Data Communications and/or the HP Communications Handbook (see bibliography). The Guidebook has valuable information on how communications equipment works and its various varieties. The Handbook has more specific information relating to HP products (be sure you have the 4/81 or later version). If you have all your equipment and HP software in place, you can now begin to write your application. It will help if you code the entire application using a programatically controlled terminal to simulate the communications link. In that way you will know that the rest of the code works before stepping into the unknown. You will have to use SPL or an SPL subprogram to establish the communications interface.

Most CS/3000 intrinsics have direct equivalents in the file system (see figure one). Treat the communications line like another file except use the CS intrinsics instead. There are several differences that you must take into account. The COPEN has parameters describing the communications environment that are quite different from the parameters for FOPEN although some of them are similar. The COPEN intrinsic format is shown on page I-30 of the Communications Handbook Version 4/81. Either the formal designator or the device parameter is required. The rest are optional. The three options fields and some other parameters are explained on pages I-13 and I-29.

IBM Bisynch requires that each series of messages be terminated by an EOT. When writing to the line (CWRITE) you must end with an EOT before you finish or can read (CREAD) data. This is done with a CCONTROL (linenumber, I, param). When reading, a CCG condition means that an EOT has been recieved instead of data. You must keep posting CREAD's until you get the CCG condition.

Error handling needs to be more elaborate than normally used for the file system. The 200-250 series of error numbers are particularly important because they may indicate problems on the line and/or problems at the other end of the line. In particular you may see many 205, 207, and 217 errors. 205 occurs during CWRITE and means that the other end is trying to send data. This occurs if you are the secondary station (the primary station, normally the IBM host, has priority). A CREAD should immediately be done to receive this data. A 207 typically ocurs during a CWRITE if the line takes several "hits" or the computer at the other end goes down. A 217 error is the result of a CREAD failure when the other end goes down. Most other errors in the 200-250 series can be corrected by repeating the operation. Other errors are hardware or software prolems at the local site. They are classified as to probable cause on page I-21 of the Communications handbook.

CS/3000 INTRINSICS AND COMMANDS

Intrinsic (Command)	Similar File Intrinsic	Comments
COPEN	FOPEN	Returns line number similar to file number
CCLOSE	FCLOSE	
CREAD	FREAD	
CWRITE	FWRITE	
CGETINFO	FGETINFO	
CGROUPINFO	None	Supplies information about a remote group on a multipoint line
CCHECK	FCHECK	
PRINT'LINE'INFO	PRINT'FILE'INFO	
CCONTROL	FCONTROL	
CPOLLIST	None	Creates or updates a poll list for multipoint lines
IODONTWAIT	IODONTWAIT	Intrinsic is the same and has the same purpose (complete I/O)
IOWAIT	IOWAIT	Intrinsic is the same and has the same purpose (complete I/O)
CLINE (Command)	FILE	
CRESET (Command)	RESET	
SHOWCOM (Command)	None	Shows line errors since last COPEN

FIGURE 1

If a large number of errors in the 200-250 series occur and the link and modems seem to be working correctly, one or more entries in this MISCARRAY parameters of the COPEN should be analyzed and possibly changed. Type 5, number of error recovery retries, may be increased from its default of 6 if it is known that the line may be poor and the application can afford the increased time required. There are five different timeouts that can be changed:

1) The receive timeout occurs only for a CREAD when the remote does not send text, an EOT or a TTD (temporary text delay) after the first text message is received or the local 3000 missed it. In the latter case, you may have to run in the BS queue to solve the problem. The default of 20 seconds seems more than adequate. It could be shortened under most conditions. If it is set at too short a time, you will get many 209 errors on the CREAD's.

2) The local timeout has nothing to do with the remote or the line. If you will be waiting for data from the remote for long periods of time it should be disabled. It is used to prevent one session or job from monopolizing the line. This timeout occurs when no CS intrinsic is activated during the time period. The default is 60 seconds. If it occurs (error 155) it will disconnect the line.

3) Connect time is the time that the CS system will wait for the modem to indicate that it is ready (DSR or Data Set Ready Line comes on). It's default is 900 seconds. It should be set to about 15 seconds in situations where it is known that the modem and line are operational. If it occurs (error 151) it will disconnect the line. If the line is disconnected it should be CCLOSE'd and COPEN'ed to continue.

4) Response timeout is the amount of time the system will wait for a response to a local action during a CWRITE. Its default is 2 seconds for a primary contention station and 3 seconds for a secondary contention station. An error 207, driver retry exhausted, will be received if this timeout occurs.

5) The line bid timeout, (error 217) occurs durng CREADS. It means that the remote has no data to send. The default is 60 seconds. The timeout may or may not indicate a true error, depending on whether data is expected or not. All these timeouts are described on pages D-12 and D-13 of the Communications Handbook (the line bid timeout is there called the "Wait timeout"). Note that they are described as they apply to RJE, but the description is the best we have seen in HP documentation.

An important part of the COPTIONS field is the "local mode" subfield, settings 0-4 refer to various options using Bisynch or SDLC, setting 5 and 6 refer to HPDLC. If this subfield is not compatible with other COPEN parameters, particularly the protocol subfields of the AOPTIONS parameters, an open failure will result.

The protocol subfield of the AOPTIONS parameter must be compatible with other options of the COPEN and the download file (or driver if the INP is not being used). If contention mode Bisynch (protocol) is desired the download file is CSDBSCO, which is the default. In other cases you will have to be careful to match the download file to the protocol. This may require some investigation. All download files start with CSD, are 515 words long and reside in PUB.SYS.

Like the file system, in the normal mode of operation the CS/3000 intrinsic does not return control to the program until the I/O is complete. By setting bit 15 in the AOPTIONS parameter to 1 the program will gain control before the I/O operation completes and the program must issue an IOWAIT or IODONTWAIT instr.insic to complete the operation. There are several differences in the way this operates in CS/3000. There are no special IOWAIT and IODONTWAIT instructions for CS/3000. Because the line numbers returned by COPEN and the file numbers returned by FOPEN are mutually exclusive, an IOWAIT with a first parameter of 0 can be used to wait on the first I/O that completes, file or line. Secondly, several I/O requests (up to a maximum of 14, 7 reads and/or 7 writes on the INP) to the same line number can be made without issuing intervening IOWAIT's.

The NUMBUFFERS parameters of the COPEN is used to set the number of I/O's that may be queued (negative value) or the number that may be queued and buffered (positive value). If you use buffers, you do not have to do the COPEN in privileged mode to do concurrent (no-wait) I/O. Having several CREAD's outstanding for a line number may be convenient (remember that the EOT requires one), but it doesn't seem appropriate when doing CWRITE's because of the difficulty taking action for the appropriate record on errors in transmission. It could be done if the sequence of records transmitted were not important and the data was kept available for re-transmission. Each CREAD and CWRITE must eventually be paired with an IOWAIT or IODONTWAIT that shows completion unless I/O is aborted by a CCONTROL with 0 as the first parameter. If the CCONTROL returns an error code of 64, an IOWAIT must be issued because the I/O has already completed.

There are several aids that can be used to help diagnose problems. For the INP the program DSM.PUB.SYS can be run to test the HP hardware, the download files and, to a limited extent, the associated communications equipment. Its operation is described in the INP Diagnostic Procedures Manual. Part No. 30010-90002.

There is also a CSTRACE facility that can be invoked by bit 2 of the COPTIONS parameter or the CLINE command. It writes to a file describing in detail every action taken by the communications subsystem. These records may then be printed by CSDUMP.PUB.SYS. There are several options that can be used. We suggest that the first attempt to employ CSTRACE be with no options set. CSTRACE generates a large number of entries for every intrinsic called, even if no options are used.

If the programmer still has difficulty after using the diagnostic tools mentioned above, he might try to use a line monitor to see what is actually happening on the line. He may be able to borrow one from HP or rent one from a supplier.

In summary the HP communications products work well, but sometimes it is necessary or advantageous to use the CS/3000 instrinsics directly. This is possible but it requires considerable technical expertise and even more patience. Communications technology itself is complicated. The user is advised to use one of the HP products if at all feasible. Only if he has the necessary resources should he proceed with direct use of CS/3000.

Bibliography

Binary Synchronous Protocol, IBM form no GA27-3004-02

Communications Handbook, HP part no. 30000-90105
(Note: The descriptions of the CS/3000 intrinsics were removed from the 3/83 version, the 4/81 version has them.)

Computer Networks, A. Tenanbaum, Prentice Hall 1981

CS/3000, HP External Reference Specifications

Guidebook to Data Communications, HP part no. 5955-1715

HP3000 Data Communications Products - Specification Guide
HP part no. 5953-7444

HP 30010A/30020A Intelligent Network Process (INP)
Diagnostic Procedures Manual HP part no. 30010-90002

NETWORKING MADE SIMPLE

Prepared For:

HP 3000 International Users Group Meeting
Edinburgh, Scotland
2nd-7th October 1983

Presented By:

Jim Geers
Systems Networking Specialist
Hewlett-Packard Company

Introduction

Most of us have realized that for our Information Systems to become more effective in the 80's, success will depend on the ability to network all systems and workstations components into a single, shared resource environment. However, when the lines are drawn between these various components within your network plan, what types of communication capabilities should you expect? What types of networking services are, and will be, available for your applications? What is the most cost effective method for linking these network resources together today and in the future.

This presentation will address these questions and will help clarify and demystify the data communication function which is dominated by new buzzwords and acronyms. The primary methods for linking systems today will be discussed as well as the international and defacto standards which are currently being proposed.

Inter and Intra System Communications

Imagine this scenario. Your boss asks you for a report tommorrow and the information you need for this report is located on his Visicalc file, his secrertary's word processor, a graphics system and sales information systems located at the 8 major sales offices around the world. To most of us, this type of request is something we would have nightmares about at night.

How do we begin to collect all these various forms of information from various locations and integrate them to solve your business problem? There are two alternatives you could choose. One is to purchase all of your information processing equipment from one vendor. And that's assuming that one vendor can provide you with all of your processing needs and can integrate all of these forms of information. Some vendors are in a better position to provide this capability but no vendor is even close today. The second alternative is to select the best vendor for each individual processing need and try to merge the information together. This is the challenge of the multi-processor environment and there are probably very few companies today who can say that they purchase all information processing related equipment from one vendor.

When it comes to multi-vendor environments, there are two levels of problems to consider. The first is concerned with the pure transportation of information from one system to another. For example, if you have two different systems, how do you transfer information economically and effeciently while insuring that it arrives correctly? This is the easier problem to solve and there area many available solutions today such as 2780/3780 (RJE) emulation and asynchronous (teletype) interfaces. This area is also one which is being very aggressively pursued in the industry today with standards. These standards efforts will be addressed later. The second problem is much more complex due to different (and incompatible) operating systems on each system which provide different sets of functionality. For example, it would be very difficult to query a local data base and if the information is not available, automatically query remote data bases. It would be difficult because a request for information on one system will almost certainly have a completely different form and syntax on another system. Another example; once you transfer a document or graph between systems, can the receiving system understand its file structure or even display it? Probably not. The basic problem comes down to the fact that information in each particular vendor's system is stored and accessed in it's own particular form. Before we can solve any of these two levels of problems, an industry-wide standard networking architecture is necessary. Many of you have probably already painfully realized that there are no data communication standards today.

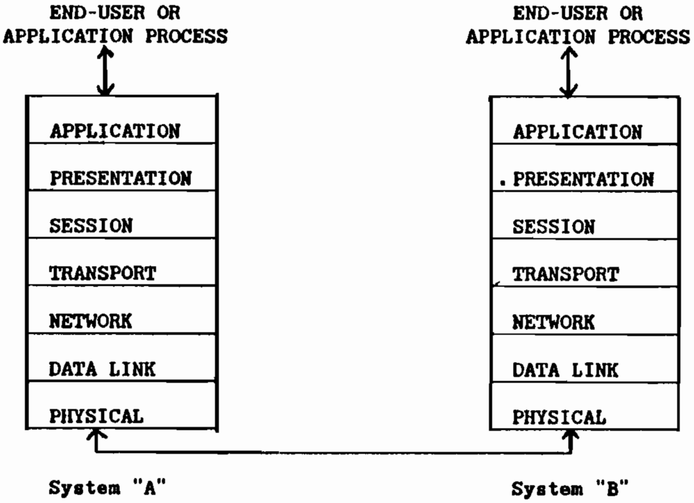
Why A Standard?

Today's vendors can support one or more networking architectures: vendor unique, IBM's SNA or the International Standards Organization's (ISO's) model for Open Systems Interconnection ("OSI Model"). If you select a vendor unique network (including IBM's SNA), you are then forced to buy all the computers and terminals from that vendor or you could spend a lot of time and money on custom interface adapters and special software. With an industry network architecture, such as the one proposed by ISO, multi-vendor networks would be possible allowing you to choose the right type of processing capability for the job, without having to compromise on a single vendor's products.

This may sound surprising but vendors, too, want a standard. A networking standard will develop a huge market, not a lot of tiny ones. Vendors make more money sharing a huge market rather than designing their the own proprietary network structures. Basically, economics is driving the ISO recommendation, not technology.

There are many additional side benifits to a common industry network architecture which will be realized in the nearer term. These will be discussed in the next section.

OSI Model


END-USER OR
APPLICATION PROCESS
END-USER OR
APPLICATION PROCESS
APPLICATION
PRESENTATION
SESSION
TRANSPORT
NETWORK
DATA LINK
PHYSICAL
APPLICATION
PRESENTATION
SESSION
TRANSPORT
NETWORK
DATA LINK
PHYSICAL
System "A"
System "B"

As mention earlier, the primary benefit of this model is that if all vendors adapted it, multi-vendor communications would be greatly simplified. The way it works is that the end-user or the application accesses the topmost layer. If data is to be sent from System "A" to System "B", the data will pass through all the layers. Each layer adds a parameter that defines the function of that specific layer down through the bottom layer until if forms a completed frame. The frame is then sent across the transmission medium and the reverse process takes place. the parameters at each layer will be stripped and the data will be presented to the end-user or application.

This model offers some additional benefits as well. First, by separating the communication function from the application, you can avoid costly conversions when the location of your data moves somewhere else. This transparency allows the user to access data in the same manner whether the information is local or remote. HP's DSN/DS is an excellent example of an application independant communications product.

The second benefit of this model is due to the layered approach. By specifying certain functions for each layer, new capabilities or performance enhancements can be integrated without affecting the way we access the data communications function. Let's look at a simple analogy to explain this.

The layered approach of sending a packet of information from one computer system to another is very roughly similar to sending a letter from one person to another. The top two layers, Application and Presentation are similar to corresponding in the English language. Both sender and receiver of the letter must speak the same language. If the sender writes in English and the receiver only understands Hindu, the message will not be completed. The same holds true for computer to computer communications. Next, the Transport layer, among other things, makes sure the data is sent to the correct node in the network and is analogous to the Post Office which controls the sending and receiving of our mail from source to destination. The Network layer deals with the routing of the data and is like the routing of our mail whether it is done by air, train, truck or foot. The Data Link layer assures that characters of data are successfully transferred from sender to receiver, just like certified mail. The Physical layer specifies the physical medium of the data transfer, similar to a piece of writing paper.

These are very rough comparisons, but they help illustrate a point. Say, for example, the Post Office has been using trucks to transmit mail and they begin to take advantage of airplanes. This faster transportation method improves overall service without affecting the way you send a letter. The day we can send mail "electronically" through the Post Office instead of using paper will also improve the communication function.

Application and Transport Services

As mentioned before, there are two level of problems with multivendor communications: the physical tranportation of information from one system to another and the incompatabilities between the operating systems. The OSI model addresses both a these problems and so we tend to group the 7 layers into two catagories. The lower layers are called "Transport Services" and are concerned with the physical transportation of data. The upper layers are grouped under the heading "Application Services" and deal with the differences in operating systems. Some examples of application services would be virtual terminal, file transfer, remote file, data base and peripheral access and program-to-program communications, all of which are available with DSN/DS on the HP3000. These capabilities can become complex even between HP3000 much less different vendor's systems. Therefore, in order to walk before we run, we are focusing our standards effort on the the Transport Services.

Transport Services Standards

To date, vendors have only ben able to agree on certain transport level services such as Public Data Networks (PDN) and Local Area Networks (IEEE 802) which are compatible with the bottom 3 and 2 layers respectively. Some vendors claim that the support of these lower layers provides multi-vendor communications. This is not true because the remaining upper layers are still vendor supplied and therefore vendor unique.

Even today's transport services are primarily vendor supplied. The Physical layer are mostly hardwired, point-to-point lines for local area communications and phone lines, both dial-up and leased, for wide-area communications. The major benfit of the telephone is its widespread availability.

This is offset by by their relatively low spead, high error rate and inefficient utilization of the line. The telephone is optimized for voice communications not digital communications. The growing market for digital communications is bringing forth new transport services such as Public Data Networks (PDN), Local Area Networks (LAN), Satellites and Private Branch Exchanges (PBX). The following is an introduction to these new transport level services and it highlights their major benefits.

Public Data Network

Also known as Packet Switched Networks, this transport service allows system to system and system to terminal communications over PDN's such as Telenet and Tymenet in the U.S. and other located in every major country in the world. PDN's provide a cost effective alternative beyond the traditional leased and dial-up telephone lines for digital computer communications. With a PDN, you primarily pay for the amount of data you send across the network and not the connection time or distance between systems.

PDN's also simplify interconnections between systems because one single connection into the PDN allows communication to multiple systems and terminals sinultaneously. Therefore, it is a much better price/performance alternative for high volume applications with multiple, geographically dispersed nodes. Another benefit is reliability. If a line fails within the network, an alternate path is automatically selected. This is not true of a leased line.

Local Area Network

An LAN can best be defined as a networking system owned by a single organization within a limited geographical area. The benefits of an LAN are that it permits resource sharing among computers and peripherals and it simplies interconnections.

Ideally you could put a single line within your facility and you can connect all information processing related equipment. The benefits are increased flexibility, resource sharing (network resources), simplifies installation (plug in the wall) and improved performance (higher speeds). As we discussed earlier these benefits cannot be achieved without standards.

Today there are several vendor defined architectures and proposals however there is a standard which exists called IEEE 802. This recommendation has been endorsed by over 20 electronic companies including HP and the original developers of Ethernet. IEEE 802 specifies an LAN with data rates of 1M, 5M, 10M and 20 Megabits per second with a range limited to 2.5 kilometers over a baseband coax cable.

What is baseband? Baseband is capable of digital transmission only. The alternative is broadband which is capable of analog transmission and therefore can incorporate voice, video and data communications. Broadband is also has a higher bandwidth and therefore offers multiple logical cables (channels) which to use. While broadband has its advantages, the major trade-off today are lack of a standard and, primarily, costs. Most organizations today cannot cost/justify a broadband network based on its capabilities today. For most organizations and applications today, baseband appears to be best suited.

Satellites

Technology advancements in satellite communications is making this transmission service a more viable alternative. Today, many organizations are finding that they can cost/justify the lease of a satellite station and channel similar to the way you lease a phone line. The benefits are significantly higher speeds and higher thru-put rates for remote communications.

Several satellite vendors also offer "broadcast" capability which allow you transmit information to multiple sites simultaneosly.

Private Branch Exchange

Today you already have and information network within your organization. Its your phone system. At the heart of your phone system is a PBX which is an intellegent switching device. You may be surprised to know that most PBX's offered today are digital switching devices because it has now become more cost-effective to convert your analog voice to digital.

If you have installed (what is referred to as) a third-generation PBX, there is no reason why you can't take advantage of existing phone lines for your digital communications! And in fact this does offer several advantages: avoids duplication of wiring, lowers cost per port (cost of new wiring + cost of union electrician), flexibility (access to multiple hosts), resource sharing (port contention), and more.

Conclusion

As we all know, it is a very competitive environment today and it will become even more so in the future. You know that the computer industry has and will continue to develop the tools to help you manage all forms of informations and make all areas of your business more productive. What you might not have considered is that within your industry, major competitors will be on the same level of computer automation in such key areas as accounting, manufacturing, computer aided engineering and office automation. Therefore, one of the key differentiating factors which will distinguish the leaders from the followers, is how effectively you exploit networking technology. Every effort today in planning to integrate all your information systems will put you into a better competitive position tomorrow.

HEWLETT PACKARD

PERSONAL SOFTWARE DIVISION 19420 Homestead Road, Cupertino, California 95014 Telephone 408 725-8111

TECHNIQUES OF LOCAL AREA NETWORKING

Marc Burch
Hewlett-Packard Company
Personal Software Division
19420 Homestead Road
Cupertino, CA 95014

ABSTRACT

The ever growing demand for cost-effective and word-processing applications is making Local Area Networks (LAN's) one of the most important communication technologies of the 80's. To meet this exploding demand, many manufactures are producing somewhat complete networks and piecemeal products with varied cost/performance capabilities. This has filled the workplace with a wide ranage of terminals, copiers, printers, telephones, plotters, personal workstations, computers, word processors, facsimile machines, private automatic branch exchanges, personal computers, and other office equipment within the same office, building or factory. We are approaching the maximum inflection point of being able to use all of this standalone and huge variety of equipment in an efficient and cost productive way. In order for groups of people in different organizations to function as efficient and well integrated units, the computers, peripherals and other equipment that they depend on must be able to communicate and exchange information quickly, easily, and reliably. An integrated total system approach based on local area networks offers a way to provide this communication exchange.

This paper provides a general introduction to local area networking basics, concepts, marketplace, business applications, and installation considerations along with reviewing issues and trends in local networking.

The potential market for Local Area Networks is very diverse and exciting. The market will be in four main areas - office environments, manufacturing plants and complexes, transaction services (Financial Distribution and Reservation systems) and educational establishments.

The largest market will be in the office environments of major corporations, where automation of office functions and increasing use of electronic storage media will require high capacity network systems with multiple channel capabilities. Every office in every company, from the smallest to the largest, in every industrial nation in the world is a potential candidate for a local area network. Most observers believe local networks will become as widespread in the office as the telephone system. Applications for intra-company networks include:

- o *Data and File Exchanges between Workstations*
- o *Peripheral Sharing*
- o *Electronic Mail*
- o *Access to Data Bases and Computing Power of a Mainframe Installation*
- o *Video Conferencing*
- o *Voice Store and Forward*
- o *Material Flow Control*
- o *Inventory and Cost Control*
- o *Engineering*
- o *Others*

In manufacturing environments, local area networks will be increasingly used to connect terminals, office data and word processing equipment and process control and surveillance systems. These environments include:

- Factory Data Collection
- Building/Environmental control and monitoring (Commercial Buildings, hospitals, schools, etc.)
- Process control, computer aided manufacturing (CAM) and Computer Aided--Design (CAD)
- Energy Managementations
- Aircraft Communications
- Photo composition
- Others

Transaction services has a great need for local area networks to help with the following applications:

- Financial Transactions
- Point-of-Sale Systems
- Reservation Systems (Hotels, Airline, etc.)
- Other

In educational establishments, local area networks will be used to connect the ever increasing number of microcomputers in use for computer education, computer aided instruction (CAI) and research purposes. However, since most high schools and elementary school budgets are small, they typically purchase only two or three disks and a few printers, and make them available to all other computers via some type of local area network or multiplexing arrangement. This will change with the development of new storage and information sources that will significantly increase the use of computer/communication networks within the school system. Other applications inlcude:

- Scientific
- Laboratory

Several different technological developments allowed the evolution of new and exciting approaches to local area networking.

First, the much heralded advances in LSI (Large Scale Integration) and VLSI (Very Large Scale Integration) technology (Computers or Microprocessor chips) made it economically possible to distribute miniccmputers and a whole array of intelligent, task-oriented peripheral equipment.

Second, many important gains in communication protocols have been combined with this very sophisitcated LSI and VLSI technology in nodes and network interfaces to provide the functions and performance levels needed for local area network communications.

Third, much more of the lower level protocols responsible for interfacing to the network and controlling network functions are being designed into the network hardware. As a direct result of this:

- *Nodes have less overhead associated with network control and now can better perform their designated functions.*
- *The network hardware components can be mass produced in volume and at lowered cost, simplifying connection to the network and encouraging greater participation by users and equipment.*
- *Communications protocols that become standardized can be incorporated into the network hardware, thus allowing a variety of nodes from many different manufacturers to communicate without the need for expensive custom interfaces, and giving users increased vendor hardware and software independence.*

The trend today is towards greater numbers of separately identifiable computer based systems, due to the decrease in prices and computer processing components within an organization. This has caused a greater awareness (at both individual and organizational level) of the benefits of convenient interconnection of systems, often supplied by different manufacturers, to achieve coordinated access both to common resources (such as databases, analysis programs, development tools and office-style memos and reports) and to sophisticated or specialized (and therefore, expensive) resources such as mainframe processor, file archive and management facilities, printers, plotters, etc. There is also a need for overall management control of system proliferation, duplication and/or dilution of effort, synchronization of activity and any other factors that can hurt an organizations resources.

Let's now look at what exactly is a local area network? Basically, a local area network is a data communications system that allows computers and peripherals to talk to each other over a common transmission medium.

What are local area network characteristics?

- *Reasonably high data transfer rates - one megabit/sec or higher. Usually, near 10 megabits/sec. Short transmission times make network operation transparent to users, allowing fast, multiple-station access.*
- *Limited geographical coverage - They usually have a diameter of a few kilometers, thus allowing rapid access and communication among distant organization groups.*
- *Low transmission error rates - Networks reliably accommodate heavy data transmission traffic should an error occur, a network station can detect it and institute a recovery.*
- *Low-cost connection and installation - You can attach or delete stations in the network without operational charges. Connection cost shouldn't exceed 10 to 20% of station-equipment costs.*

- *Large number of users - Networks can support tens of thousands of users and don't constrain organizational growth.*
- *Reliability and Availability - Networks can remain unaffected by individual failures or removals for service.*
- *Security - You can restrict network access to confidential, classified or sensitive data file.*
- *Flexible Topology - You can modify a network as the organization expands.*
- *Multimedia Communications - Some networks handle voice and video communications as well as data.*
- *Multivendor Compatibility - Networks can contain equipment from different vendors which allows greater functionality.*
- *Single Organization Ownership - Networks are usually owned by one organization and are designed by the organization to satisfy its needs. Gateways are used to communicate with other organizations.*

The design elements of a local area network must be carefully analyzed in order to provide the required levels of data communication capabilities and network performance. Anticipated use usually will determine which network type best matches your application requirement along with the following questions:

- *How access to the network and message traffic will be controlled*
- *How many and what kinds of nodes can participate and where they can be located*
- *How nodes will interact in the local area network - with other nodes and other networks*
- *The performance dynamics of the network - speed of response, ability to handle traffic loads*
- *The hardware and software that will be needed to implement the network, and all associated costs thereof*
- *The network applications that will be possible.*

However, before you can do your network and application planning, you need to understand the following:

1) *Local Area Network Topology*
   Point-to-Point or Multipoint

   A. *Ring*
   B. *Star*
   C. *Bus*

2) *Channel Access*

   A. *Polling*
   B. *Contention*

3) *Local Area Network Transmission Media*

   A. *Twisted-Pair Wire*
   B. *Coaxial Cable*
   C. *Fiber Optics*
   D. *Unbounded Medium (Radio, Microwave, and Infrared)*

4) *Signalling Techniques*

   A. *Baseband*
   B. *Broadband*

1) Local Area Network Topology

- *There are two kinds of links that serve as the building blocks of network topologies. Point-to-point and multipoint or multidrop. A Point-to-point links is a circuit which connects two (and only two) nodes without passing through an intermediate node. A multipoint or multidrop links is a single line which is shared by more than two nodes. Multipoint lines can be used to reduce the number of lines required to connect nodes and to reduce line costs.*

A. *RING TOPOLOGY*

The distinguishing feature of ring topologies is that nodes, which are connected by point-to-point links, are arranged to form an unbroken circular configuration. Transmitted messages travel from node-to-node around the ring. Each node must be able to recognize its own address in order to accept messages.

In addition, each mode serves as an active repeater, retransmission messages adddressed to other nodes.

A. *Ring Topology (Continued)*

The need to retransmit each message can make ring nodes more complex than the passive nodes on a bus network. When Ring configurations are used to distribute control in local networks, access and allocation methods must be used to avoid conflicting demands for the shared channel. One way this is done by circulating a bit pattern, called a token, around the ring.

A node gains exclusive access to the channel when it grabs the token. It passes the right to access the channel (i.e. the token) onto other nodes when it is finished transmitting. Rings provide a common network channel wherein all nodes are fully connected logically. When control is distributed, each node can communicate directly with all other nodes under its own initiative.

Ring networks with centralized control are often called Loops. One of the nodes attached to the network controls access to and communication over the channel by the other nodes. Once a node is permitted by the control node to transmit a message, it can travel around the ring to its destination without further intervention by the control node.

In configuring ring networks, rings must be physically arranged so that they are fully connected. Lines have to be placed between any new node and its two adjacent nodes each time an addition is made. Thus, it is often difficult to prewire a building for ring networks in anticipation of nodes to be added in the future.

Failure of a node or an active component, adding a new node, or any other break in the ring confugration will almost always cause the network to stop functioning. Steps can be taken to allow bypass of failure points in distributed rings, although this usually increases the complexity of the repeater at each node, as well as the component costs. Failure of the control node in a centrally controlled ring would inevitably lead to network failure as well.

B. *STAR TOPOLOGY*

The distinguishing feature of Star is that all nodes are joined at a single point. Star configurations are frequently used for networks in which control of the network is located in the central node or switch. Point-to-point lines connect the central and outlying nodes, thus eliminating the need for the complex link and control requirements of other topologies.

A Star network could be constructed so that the control of communications would be exercised by one outlying node, or distributed generally to all outlying node, or distributed generally to all outlying nodes. In either case, the control function of the central node would be minimized. The node would serve as a simple switch to establish circuits between outlying nodes.

In all Star networks the central node is a single point of network failures. If it goes down, so does the entire network thus, reliability and the need for redundancy measures are important issues.

B. Star Topology (Continued)

The Star is often used in timesharing applications, in PBX (Private Branch Exchange) telephone networks and in small clustered networks like word processing clusters.

C. BUS TOPOLOGY

The Bus topology functions in the same ways as a multipoint line and bus nodes share a single physical channel via cable tape or connectors. Messages placed on the bus are broadcast out and nodes must be able to recognize their own address in order to receive transmissions, however, unlike nodes in a ring, they do not have to repeat and forward messages. Therefore, there is none of the delay and overhead associated with re-transmitting messages at each intervening node, and nodes are also relieved of network control responsibility at this level.

Bus networks are easily configured and expanded and the network operation will continue in the event of node failures. Distributed Bus networks have been around for some time. This is attributed to the fact that the transmission media (usually coaxial cable or twisted pair wire) and transmission have been in use for some time by the cable and TV industry and for telephone and data communciations.

There are many considerations for the design of Bus networks. Components, such as transmitter/receivers, must be designed for reliable and mainframe operation. There must also be test equipment that will allow for fast and accurate Bus Fault detection and isolation to facilitate repair and maintenance.

2) CHANNEL ACCESS

A. POLLING

Polling Techniques determine the order in which nodes can take turns accessing the network, specifically so that direct conflict (i.e. collisions) between nodes is avoided. Polling is thus referenced to as an non-contention method of network access.

Centralized polling may be based on a polling list with an arbitrary order (A,C,B,E,D..) or the order could reflect network node and traffic priorities. The order of access could also be based simply on the physical location of nodes.

Distributed Polling also allows control of access to the network by either token passing, slotted ring, or Cambridge ring. Token passing is a mechanism whereby each device, in turn and in a predetermined order, receives and passes the right to use the channel. Slotted rings emply a number of slots or frames of fixed size circulate around the ring. If a node chooses to transmit, it waits for a free or unused slot, inserts data into the appropriate field, and indicates the source and destination address. As in token passing, nodes along the way check to see if the frame is addressed to them. Due to the high speed of slotted rings certain ineffcicencies exist, such as only several bytes of data can be

A. Polling (Continued)

placed in each frame. Therefore, frames contain a high amount of control information vs data.

The Cambridge ring system is based on the establishment and use of circulating slots or packets of fixed size which are successively accessed, filled and read by network nodes as they pass by. Each packet slot travels in one direction only and makes a complete revolution of the ring.

B. CONTENTION

Carrier sense multiple access with collision detect (CSMA/CD) anticipates conflicts or collisions and is designed to handle them. The multiple access feature of CSMA/CD allows any node to send a message immediately upon sensing that the channel is free of traffic. Therefore, you do not have the substantial portion of the waiting that is characteristic of non-contention techniques. This access control methods chief advantages lie in its simplicity reflected in lower cost per node because the scheme needs no complex priority access circuits - and its variable length message handling efficiently.

Carrier sense is the ability of each node to detect any traffic on the channel. Nodes will not transfer if they sense that there is traffic on the channel. Nodes will not transfer if they sense that there is traffic on the channel. However, collision could occur betwewen two messages due to the propagation delay (time it takes for the signal to travel across the network). As both nodes thought they had an open channel.

After detecting a collision, each node involved backs off, waits, and transmits again. This waiting period is usually random as it has proven to be more effective in avoiding further collisions.

The most efficient use of CSMA/CD is when the packets are larger and therefore you have fewer collisions.

Several Bus networks used collision avoidance instead of CSMA/CD due to their lightly loaded applications. Collisions get detected by the nodes sending and receiving circuits. The sending node waits for an acknowledgement signal. If it does not come, it will retransmit until successful.

3) LOCAL AREA NETWORK TRANSMISSION MEDIA

A. Twisted-Pair-Wire - Was one of the first wire types used in telephone communications and that is still true today. Pairs of wires are twisted together to minimize the interference created when adjacent pairs of wire are combined in multipair cables. Wire is usually made of copper and is inexpensive and easy to install. Used mainly in low speed data equipment with the average in the range of 9.6 Kbits per second. However the wire does emit and absorb high amounts of electrical interference, which can cause error rates.

Twisted pair provides a good media for integration voice and data through Digital Branch Exchanges (DBX) or Computerized Branch Exchange (CBX).

B. Coaxial Cable - offers large bandwidth and the ability to support high data rates with high immunity to electrical interference and a low incidence of errors. Because it maintains low level capacitance in lengths to several miles, coax allows high megabit per second data rates without signal refeneration, echoes or distoration. Coaxial cable used for CATV (Community Antenna TV) has bandwidth in the range of 300-400 MHz.

C. Fiber Optics - Although expensive, possesses inherent local network point-to-point performance capabilities that outclass all other transmision media. Currently available fibers have usable bandwidth of up to 3.3 billion Hz, data rates of over one G-bits per second, high voltage isolation, non-electronic radiation, small size and light weight, and error rates are very low (one bit error per 109 bits).

However, fiber optics are still too costly and are also very hard to tap into to add additional nodes.

D. Unbounded Medium - Radio, Microwave and Infrared

Today, there are a few commercially available local area networks based on these unbounded medium technologies. Given time and the advancement of technology these three major types of signals will undoubtedly become more prevalent.

4) SIGNALING TECHNIQUES

a) Baseband

- Lower cost of cable used for interconnection
- Baseband interfacing is less expensive than broadband modems
- Reasonably high data transmission rate
- Acceptably low error rate

Baseband local area networks are predominantly used for data communications and therefore the signals and transmission technology implemented are digital (optimized for data) with maximum capacity of approximately 10 MBPS. Most often 3/8 inch coaxial cable is used with Bus Topology with contention control - usually CSMA/CD.

b) Broadband

- High total bandwidth allows transfer of data, voice, video.
- Total bandwidth can be (statically) subdivided into many independent channels for transparent use by attached devices.
- The technology is common to CATV (Community Antenna TV), affording some cost-benefits with conventional CATV services.
- Acceptably low error rate for most applications.
- No significant geographical extent limitation (up to 500 Km).

The advantages of broadband are to be found in applications which call for many point to point connections where the interconnection pattern is essentially static or where very high point-to-point transfer rates (such as video) are required.

A broadband network uses analog transmission techniques and can accommodate up to 500 Mbps of information. Broadband networks use frequency division multiplexing (FDM) to divide a single physical channel made of 1/2 inch 75 OHM CATV coaxial cable into a number of smaller independent frequency channels. These smaller channels can be allocated different bandwidths so that they can be used to transfer different forms of information - specifically voice, data and video. Time division multiplexing (TDM) of the total cable capacity is a more complex process and is becoming available due to the declining cost of digital logic.

Broadband shortcomings are that in addition to requiring network and station interfaces, it uses expensive fixed - frequency or frequecy agile (Tunable) modems costing $500 to $1200. In addition to easily doubling broadbands interface cost, tunable RF modems prove difficult to check, maintain and adjust because they are usually installed behind walls and above hung ceilings.

b) Continued

Broadband networks must also rely on a central transmission facility or head end in a single cable network. This facility acts as the networks technical control center and filters incoming RF signals. But it also represents a point that could deactivate the entire network, if it fails.

The lack of industry wide accepted standards for interfacing equipment to broadband network is still a disadvantage. However, for Baseband this is not true as many vendors have endorsed the IEEE802 standard for Baseband networks.

In the final analysis, the choice between Baseband and Broadband is, and will continue to be, dictated by engineering economies.

PRIVATE BRANCH EXCHANGE (PBX)

The PBX is another solution thats often easier to install and sometimes less expensive then local area networks. Although first and second generation PBX's were geared for analog voice communications, third generation PBX's or CBX's (computerized branch exchange) and DBX's (digital branch exchange) use digital signal-processing techniques to channel digital data and voice. CBX's have three basic characteristics; distributed or centralized architecture, integrated voice and data (up to 56 kbit/s), and non-blocking (uses fixed-time slot assignment to allow 100% of the installed devices to be simultaneously active) configuration.

There are many different trade-offs between third-generation CBX's and local networks. The applications to be interconnected and the workplace site considerations will dictate the final implementation.

FUTURE ISSUES FOR LOCAL AREA NETWORKS

Shared Resources - The proliferation of inexpensive intelligence is encouraging a drive towards shared resources rather than to shared logic. The number of individual devices that need to communicate will grow explosively over the next few years.

Wire-Less Connection and Satellites - Terminal equipment will be able to communicate with the local area networks by means of Radio links and Infrared tranmissions.

Standardization - Starting to see some of this now (i.e. IEEE802 standard for Baseband) but the industry needs higher level standards to be set. The timescale for a satisfactory outcome of the various standardization activities is most likely to be on the order of 3-5 years. The International Standards Organization (ISO) is still working on the Open Systems Interconnection Standardization (OIS) model. This OIS model is a seven-layer network that deals with communication functions ranging from the physical characteristics of networks to message transmission and application chores. The bottom three layers ((7) physical layer, (6) data-link control layer, (5) network layer) are specific to local networks, while the top three ((3) Session layer, (2) presentation layer, (1) application layer) are common to all systems. The middle layer ((4) transport layer) resolves physical differences between networks.

Future Issues (continued)

Hardware Trends - Downward for prices as the combination of economics being achieved in packaging of electronic logic (LSI, VLSI, etc.) Along with new developments in information transmission media (Fiber Optics, Infrared) will ensure local area networks strong future.

Network Management - The growing dependence of large numbers of systems to local area networks will demand guarantees on networks availability and performance. This will have to be provided through effective resource management and planning.

Network Security - Security vulnerabilities increase as you grow your communication links. Therefore, its important that management sets up a control strategy to consider data security, risk and vulnerability assessment, access authorization and data encryption.

SUMMARY

The future for Local Area Networks looks exciting and wide open. Over the next two or three years considerable progress will be made in the development of local area networks. This is necessary if we are ever going to have the much heralded "office of the future". These developments would not just be to benefit the office but also where information is being handled - process industry, manufacturing, teaching, research, etc. However, the real challenge lies in being able to develop new applications that will take advantage of Local Area Networks.

REFERENCES

1. *Hopkins, G. and Meisner, N. "Choosing Between Broadband and Baseband Local Networks," Mini Micro Systems, June 82 pp 265-274*

2. *Kinnucan, P., "Local Networks Battle for Billion Dollar Market", High Technology, November/December 1981, pp 64-72*

3. *Heard, K. Local Area Networks, Gartner Group Special Report, Feb 1982*

4. *Digital, Introduction to Local Area Networks, 1982.*

5. *Kotelly, G., "Local Area Networks - Technology", EDN, 1982*

6. *Strategic Incorporated, Intra-Company Networks: Broadband vs. Baseband, the Key Issues, February 1982.*

7. *Systems and Software, Upper Level Protocols, March 1983.*

"How HP Networks Can Improve Your Productivity"

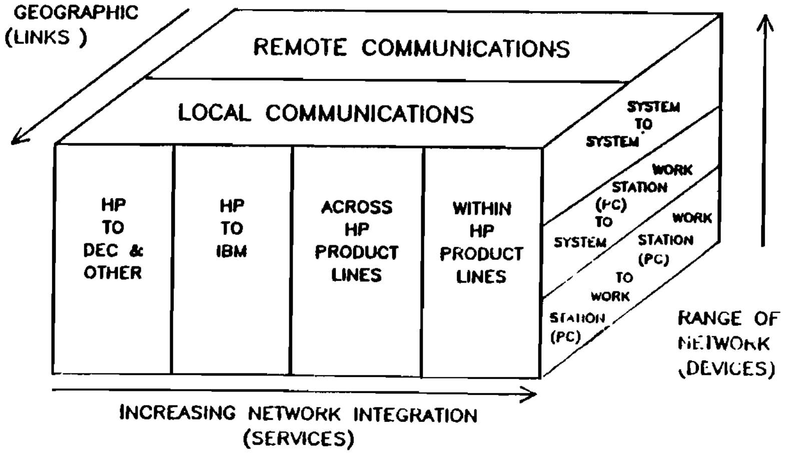
Jay Kidd

DSN/DS Product Manager
Information Networks Division
Hewlett Packard
19420 Homestead Road
Cupertino, CA 95014

This presentation will cover Hewlett-Packard's networking strategy and how it will improve your productivity. Information on upcomming products and longer-term networking directions will be presented with the intention of clarifying the capabilities and positioning of the HP offerings.

The sensitive nature of some of the information to be covered prevents the possiblity of providing a written presentation before the conference.

HP's DATACOMMUNICATION PRODUCTS
HP/DSN


GEOGRAPHIC
(LINKS)
REMOTE COMMUNICATIONS
LOCAL COMMUNICATIONS
HP TO DEC & OTHER
HP TO IBM
ACROSS HP PRODUCT LINES
WITHIN HP PRODUCT LINES
SYSTEM TO SYSTEM
WORK STATION (PC) TO SYSTEM
WORK STATION (PC) TO WORK STATION (PC)
RANGE OF NETWORK (DEVICES)
INCREASING NETWORK INTEGRATION
(SERVICES)

DEVELOPMENT OF THE BRITOIL INTEGRATED TERMINAL NETWORK

N H SHELNESS - BRITOIL plc

1 Introduction

In 1978 The British National Oil Corporation (BNOC) undertook an evaluation of small computer systems with a view to providing local terminal based processing power at each of its three main business centres (Glasgow, Aberdeen and London). As a result of this exercise HP3000s were selected and a communications architecture based on DS/3000, MRJE/3000, MTS/3000 and directly connected terminals was defined (figure 1). While this configuration was never installed, the use of DS/3000 and MRJE/3000 has continued to this day (see figure 2), while the use of MTS/3000 and directly connected terminals was soon rejected. This paper discusses the reasons for this rejection and the alternative terminal connection architecture that has evolved in its place.

2 Early Growth

In late 1978 BNOC installed a half megabyte HP3000 Series III in its 220 St Vincent Street, Glasgow office to serve the terminal processing requirements of its Glasgow, Aberdeen and London staff. At this point BNOC had staff and terminals sited as follows:

	Staff	Terminals
Glasgow	448	24
Aberdeen	276	6
London	317	4

Figure 3 depicts the connection architecture in place at that time.

By mid 1980, BNOC had installed an additional 2 x 1 megabyte HP3000 series IIIs and upgraded the initial system. One of these new systems the first was sited in its St Machar House Aberdeen office, the second was installed along side the first in Glasgow. By this time BNOC has staff and terminals sited as follows:

	Staff	Terminals
Glasgow	948	60
Aberdeen	801	22
London	177	11

Figure 4 depicts the connection architecture in place at that time. It had become clear that the policy of using directly connected terminals and MTS/3000 was not going to be very effective in BNOC's environment.

3 Limitations of Direct Connections

Direct connection, using only a cable to attach a terminal to HP3000, is by far the simplest and cheapest means of terminal attachment. If there is only a single HP3000 at a location and terminals are housed within the same building, no other form of connection need be considered. But by 1980, BNOC had two HP3000s in a single location with the possibility of one a "development" system being used as a standby for the other, a "production" system. This was not possible. For while DS/3000 allowed terminals directly connected to the "production" system to access the "development" system while both were operational, terminals directly connected to the former could not access the latter as a standby system when the former was down.

4 Limitations of MTS/3000

Unlike direct connection, MTS/3000 in 1980 posed a number of intrinsic problems in addition to those that resulted from specifics of the BNOC environment.

The intrinsic problems were:

1 atrocious performance (output to a single printer could cripple response)

2 fault finding was extremely difficult

3 expensive rapid poll modems were required

4 a maximum line speed of only 4.8 kbps was available

5 it was difficult to add or remove terminals

6 only 2 MTS circuits could be supported on each of our systems

While a number of these intrinsic weaknesses (1.4 and 6) were later removed by new HP hardware (the INP which replaced the SSLC), the rest have remained.

Those problems that derived from the BNOC environment were caused by rapid organisational growth. Not only was BNOC growing rapidly, but so were individual departments with the result that they rapidly outgrew their own accommodation. This led to a situation in which new offices were being rapidly acquired and departments were moved (on average twice a year). This in turn resulted in a constantly moving terminal population. For example a large number of directly connected terminals could be moved to another building at short notice. MTS/3000 could not cope either with the number of remote terminals, the number of remote sites and the rate of change. An alternative approach to remote connection was required.

5 Selecting an Alternative Terminal Connection System

BNOC in 1980 thus approached a number of data communications vendors with a view to acquiring a system that would overcome the immediate problems identified in Sections 2 and 3 above as well as form the basis of all future terminal attachement requirements. All of the vendors bid systems based on statistical multiplexors tied to cross-connection units in Aberdeen and Glasgow, though there were considerable differences in their detailed proposals.

Vendors bid systems in which the statistical multiplexors would be separate from, as well as integrated with, the cross-connection switches (see figure 5). Similarly vendors bid systems requiring manual (plug and socket patching), central operator or user controlled, cross-connection.

In the end the choice was a relatively easy one as the system that was clearly most cost effective was also joint or individual "best" in almost all categories. The choice was CASE DCX. At the time it offered the following "best" technical characteristics.

1 A common set of cards across all units.

2 The ability to support multiple composite links on a remote multiplexor.

3 Direct connection of composite links to the cross connect switch without prior de-multiplexing.

4 A simple field upgrade path from a simple muliplexor, to an operation controlled cross connect switch, to a terminal-user controlled cross connect switch to a tandem cross connect switch.

5 Multiple destinations for a single named service.

6 Automatic route selection between switches.

Though the system that was initially installed exploited only a few of these features (1, 2 and 3), the existence of the others, coupled wth continous, upward compatible, product enhancement has allowed the system to develop in the ways that will be described in the remainder of this paper.

6 The Initial BNOC Terminal Connection System

A decision was made to install the simplest DCX system possible consistent with eliminating the problems enumerated in sections 2 and 3. This involved the use of only two types of unit and three types of plug in cards. These were:

Units

DCX 830 - an upto 12 way multiplexor
holding 1 x ARQ, 1 x BUF, 1 - 3 x LSC
DCX 840 - an upto 255 way cross connect switch
holding 1 - 15 x ARQ, 1 - 4 BUF, 1 - 60 x LSC

Cards

BUF - a buffer memory card
ARQ - an N channel/single port composite link interface
LSC - a 4 channel/4 port asynchronous device interface

Upon vendor advice, only 8 x 2.4 kbps asynchronous terminal channels were initially multiplexed over a single 9.6 kbps composite link. Thus remote sites with from 1 - 8 terminals were equipped with a DCX 830 containing 1 x ARQ, 1 x BUF and 1 - 2 x LSC cards, while remote sites with 9 - 16 terminals (no site at that time had more than 16 terminals) were equipped with a DCX 840 containing 2 x ARQ, 1 x BUF and 3 - 4 LSC cards. In the case of an 830 the cross-connections between an asynchronous device (terminal) channel and a composite link channel was fixed. In the case of a DCX 840 the cross-connections between an asynchronous device (terminal) channel and a composite link channel had to be established by an operator, either locally from the 840 front panel or remotely over a composite channel from another (in this case central) 840 front panel.

Each of the 2 central (computer) sites were equipped with a DCX 840 containing enough ARQs to interface the attached remote sites and each other (the 2 central DCX 840s were linked), 3 x BUFs and enough LSCs to interface all of the available HP3000 ATC ports and all of the previously directly connected terminals. As at that time, a single terminal required semi-permanent connection to only a single HP3000, operator, front panel established, cross-connections on each central DCX 840 could effect the connection of local or remote terminals to local or remote ATC ports.

The linkage of remote DCX 830s and 840s to a central DCX 840, depended upon distance. If the remote DCX unit was in the same telephone area as a central unit, 2 x BT EPS8 circuits were installed. One (remote DCX 830) or both (remote DCX 840) of these circuits were then interfaced to the ARQs at each end by inexpensive (£400) 9.6 kbps base band modems. If the remote DCX unit was outwith the same telephone area, one (DCX 830) or two (DCX 840) BT EPS25a circuits, with PSTN dialled backup, were installed and interfaced to ARQs at either end by expensive (£4000) 9.6 kbps conventional modems. Within this configuration the number of terminals at a remote site could expand or shrink (between 1 and 16) and new remote sites could be accommodated (using the PSTN until the appropriate circuits were installed) at short (1 - 2 weeks) notice. All of the limitations of MTS/3000 had been overcome, and the "production" terminals could be connected to the "development" HP3000 when the latter hosted the "production" service.

It might appear, that with all of our objectives having been met, the end had been reached. As the remainder of this paper will make clear it was only a beginning.

Figure 6 depicts the original DCX network (of 1981).

Note: that the existing MTS circuits were not immediately withdrawn from service (our inate conservatism), and that the Glasgow central computer site had moved to Cadogan House.

7 Further Growth

Early in 1980 a decison was made to install our own IBM 3033, as an alternative to our by then massive use of a bureau service. This system was installed in Glasgow in October 1980. It had been policy within BNOC (on cost grounds) to limit the use of the bureau service to batch working. With the installation of our own system, and a policy commitment to the use of IBM based fourth generation software faciltіes (FOCUS), a requirement for joint IBM and HP3000 access from a single terminal arose. This coupled to a demand for more HP3000 terminals without a corresponding growth in HP3000 ATC ports necessitated an upgrade to our DCX network. This was almost trivially accomplished, though carefully planned for and introduced.

The 2 central DCX 840s were upgraded to DCX 850s by the addition of a single (level 1) User Switching Option (USO) card. The USO replaced operator effected semi-permanent cross-connections with its own temporary cross-connections, established for the duration of a session, after a dialogue with a terminal user at the commencement of each session. To operate this scheme, sets of channels connected to 3 x HP 3000 and IBM 3033 ports were grouped into 7 numbered USO services. These were:

- IBM 3033/3705 ports at 1200 bps.
- HP3000 "production" HP264X terminal ports.
- HP3000 "production" non HP terminal ports.
- HP3000 "development" HP264X terminal ports.
- HP3000 "development" non HP terminal ports.
- HP3000 "operations" HP264X terminal ports.
- HP300 "operations" non HP terminal ports.

Each inactive terminal channel was initially cross-connected to a USO "initial connection" channel. When the user hit the break key, the USO re-cross connected the terminal to one of its multiple "dialogue" channels. The USO then prompted the user to enter the number of the desired service, and if one of the channels assigned to the service was free (not cross-connected) the USO again re-cross-connected the terminal channel to the free channel.

The cross-connection once established was identical to an operator established (840) or hard wired (830) cross-connection.

The number of terminals, remote sites, and computer ports continued to grow, so that by early 1982 the number of channels required to pass through the Glasgow DCX 850 had outstripped the capacity of a single DCX 850. Thus a second DCX 850 was installed in Glasgow and a second simple DCX functional upgrade was effected. The level 1 USO cards were replaced by level 2 USO cards (this was effectively a software upgrade). Whereas a level 1 USO was able to establish cross-connections between terminal channels and service channnels on the same DCX 850 unit.

A level 2 USO was able to establish, through cooperation with other level 2 USOs in other 850s, all of the cross-connections: initial, intermediate, and final; to connect a terminal channel on one DCX 8 to a service channel on another, via multiple intermediate 850s. The mechanisms by which this was achieved, though relatively simple, are beyond the scope of this paper.

It thus became possible to split the Glasgow computer ports and remote composite links between 2 x DCX 850s with a resultant increase in reliability, yet at the same time all of the computer ports on all three DCX 850s became available from any terminal. Previously Glasgow computer ports had to be semi-permanently cross-connected to Aberdeen for access from Aberdeen terminals and visa versa.

The upgrade from level 1 to level 2 also brought other advantages. These were:

- service names rather than numbers
- comprehensive composite link usage statistics
- alternative routes between 850s if possible

The first was introduced immediately, the second allowed the number of terminals multiplexed to each composite link to be based on measured link utilisation rather than rule of thumb, while the third offered scope for future developments.

In the same time frame BT had granted approval for base band modems to operate at 19.2 kbps, as ARQs were rated upto 24 kbps all local composite links had their speed upgraded.

Figure 7 depicts the network after the upgrade to 3 x DCX 850s in early 1982.

8 IBM Full Screen Terminal Access

With the installation of the IBM 3033 in October 1980, a second, if in some senses unrelated, problem had arisen. For while it was possible to attach asynchronous ASCII devices to the 3033 via the DCX network and a 3705 Communications Processor, this did not provide for full screen (IBM 327x) type working, in the same way that a network of non HP block mode compatible VDUs could not be used with VIEW/3000. We were thus faced with a dilemma. Would we have to install a second parallel network of 327s with all that would entail ? The clear answer from BNOC's computer department management was no. We were not going to get into a situation in which users had two or more terminals on their desks, and so BNOC's "single terminal policy" was born. This has often been misinterpreted to imply that a single terminal type could suit all users. It did and does not. It merely indicates that a single terminal on a user's desk should be capable of accessing all of the systems, in all of the modes, that the user required. So much for the ideal. How was it to be achieved ?

The "great white hope" among CSD management at that time was a soon to be available HP product called IML/3000 (later renamed IMF/3000). This purported to allow an HP terminal to "pass through" an HP3000 as an emulated IBM 3277. Examination of IML/3000 with original mode of HP3000 operation, (eg Block Mode) indicated that its performance was going to be too slow and idiosyncratic for general use. These judgements were borne out in a trial of the product in March 1981, and the search for an alternative began.

At that time two approaches seemed possible. These were:

- acquire a number of the commercially available 3270 protocol convertors
- construct our own, channel attached 327X emulating front end processor

While the answer would seem clearly to have lain with the first alternative, it did not. All of the protocol convertors studied were unable to support HP terminals as emulated 327xs. Each of the protocol convertors could effectively support only 4 emulated devices before their micro-processors ran out of power. As we wished to support 64 emulated ports, we would need to install 16 and each would require to be attached to its own 3705 port. For all of these reasons, the first alternative was rejected and examination of the second began.

While a number of mini-computer vendors offered an IBM 370 channel attachement capability, we believed that life would be easier if we stuck to an IBM product, and hence the IBM Series/1 was chosen as our putative front-end processor and outline systems design was begun. At the same time, we could not believe that someone faced with a similar requirement (we couldn't be that unique) hadn't done it already, so a search began. It was rewarded in the autumn of 1981 with the news that Yale University, had written such a system for the IBM Series/1, and that though unavailable in Europe, it was being marketed and supported in the USA by IBM as an Installed User Program (IUP). (As an aside, over the years, many of the IBM's best software products were written by IBM field offices of users and subsequently marketed by IBM as Field Developed Programs (FDPs) or IUPs. HASP and SPF being two obvious examples).

From the product description, the "Yale ASCII Terminal Communications System" seemed an almost ideal solution (the implementation was similar in structure to that which we had ourselves designed). It could emulate both channel and 3705 attached 327x cluster controllers. Did not run out of power supporting the logical maximum of 32 emulated 327x terminals, running at 19.2 kbps, per cluster and was desgined to support virtually any synchronous ASCII VDU (only cursor addressing, erase to end of screen, erase to end of line, and destructive character overwrite needed to be present) HP terminals included via simply constructed input and output function definition tables.

A visit to Yale was arranged, to view the product in operation and confirm that its description wasn't just marketing hype. We were extremely impressed, and upon our return, an order for a channel attached IBM Series/1 and the Yale ASCII Terminal Communications System IUP (the first in Europe) was placed. Today we have two channel attached Series/1s supporting 64 emulated 327xs, and are about to install a 3705 modem attached varient in our communications backup site to provide full screen 327x access to our backup facility in case of a disaster affecting our own 3033.

The IUP was easy to install and to configure (As an aside, each and every supposedly compatible HP terminal type requires a separate definition because of small differences) has proved extremely robust, and offers a better level of response to remote HP VDUs than could be achieved with real remote 327xs. It is not surprising that nearly 400 copies of the IUP have been installed world wide in a little over two years and the rate of installation continues to climb.

9 Consolidation

1982 not only saw the installation of the IBM Series/1s but the addition of 3 x new HP3000's (2 model 64s and a model 44) and the integration of 6 x HP1000 into the DCX network, which by the end of the year allowed 350 attached VDUs and 100 printers on 25 sites to access some 240 computer ports on any of 15 computers at 6 different sites with full terminal functionality. Yet further developments seemed appropriate. These were:

1 Introduction of a more resilient network topology.

2 Upgrading of remote DCX 840s to DCX 850s.

3 . Upgrading to higher speed terminal working.

4 Upgrading of composite link speeds.

5 Installation of a centralised network monitoring and control facility.

By late 1982, though capacity limitations had forced the installation of a second DCX 850 in Aberdeen and a third in Glasgow, the network topology consisted of two star bursts with the majority of remote terminals being semi-permanently cross-connected via DCX 840s to a single composite link and hence a single central DCX 850. If a circuit or a central DCX 850 went down, all of those terminals went down with them until manual re-cross-connection, (if not link or central DCX 850 capacity limited), attached them to a live link and or central DCX 850. Also, if a disaster wiped out a central site, despite the availability of backup systems on other sites, terminals would not be able to access them as they had to be routed through the disaster affected central site.

The solution was two fold: upgrade all remote DCX 840s to 850s and triangulate all connections through a "communications backup site". The barrier to the former was simply cost.

With a USO then listed at £6500 - CASE was approached and asked to quote on a job lot basis for 10. The price that they quoted, which has since been undercut by CASE's new low list prices (£2000 for a USO or £3300 for a complete 850 including 1 x BUF and 1 x ARQ) rendered the exercise viable, and upgrade of all remote DCX 840s to DCX 850s was scheduled for phased introduction through the first half of 1983. At the same time the second composite link (at least in Glasgow) was routed not to a central DCX 850 but to a DCX 850 in a second "communications backup site" which was in turn linked by high speed composite links to the two central sites, thus forming a triangle. The first composite link was also upgraded to higher speed working if appropriate. As configured, a remote DCX 850 will first attempt to access a service through the shortest direct "primary" route, to a central site. If there is no spare capacity on the route, a link is down, the destination DCX 850 is down, or all of its ports for the requested service are busy, it will attempt a connection through a longer secondary (indirect via the backup site) route, etc.

In this way the operational viabilility of the network is automatically maintained should a composite link or central 850 fail. As the number of dedicated (830 and 840 composite and 850 asynchronous device) channels on a DCX 850 is limited to 255 with another 255 for use as inter 850 network composite channels, the introduction of remote 850s more than doubled the effective capacity of the central DCX 850s.

Though no attempt was made to upgrade DCX 830s, provision was made for re-connecting them to the backup site either via the PSTN or a backup BT EPS8 circuit in the case of a link or central 850 failure. Figure 8 depicts the network after the upgrade.

The original DCX network had been designed for 2400 bps terminal operation (the maximum speed of the HP3000 Series III). With the incorporation of the IBM Series/1 and the new HP3000s models 44 and 64 into the network, 9600 bps terminal working became possible.

While this posed no direct technical problem, CASE had been producing, and we had been installing, a 9.6 kbps LSC card for sometime, and we had a buy back agreement for replacement of all 2.4 kbps LSCs, there was still the problem of the ENQ and the ACK.

HP CPUs make use of a propriatory, logically half duplex, asynchronous pacing protocol so as not to overflow a terminal's buffer memory. An HP3000 transmits 80 characters followed by an ENQ after which it suspends transmission until it receives an ACK from the terminal. An HP1000 transmits an ENQ and waits for an ACK after 40 characters and three times at the end of each line (the reasons for the latter are lost in pre-history). If a terminal is directly connected to a CPU, the delay introduced by sending an ENQ and waiting for an ACK takes only a few character times. If one or more slow composite links intervene, it can take many character times, thereby reducing the real speed of terminal operation by an equivalent amount (eg if a HP3000 driving a VDU at 9600 bps has to wait 80 character times, due to composite link transmissions delays, for an ACK after having sent an ENQ the real transmission rate will have been reduced to 4800 bps. There are two solutions to this problem. They are:

- Build a special LSC that responds immediately to an ENQ with an ACK

- Reduce the composite link delay

CASE have done both.

An HP specific LSC with a maximum speed of 4800 bps is available which watches for an ENQ and replies immediately with an ACK if a threshold of 5 outstanding ENQs has not been reached. The ENQ is translated into a DCX control code and passed to the other end where the terminal LSC transmits an ENQ to the terminal and suspends output until it receives an ACK. The ACK is then translated into a DCX control code and returned to the other end, so as to reduce the number of outstanding ENQs by one. There is thus no transmission delay introduced by the composite link delay.

DCX ARQs operate by sending up to 128 byte packets over a composite link as a single chunk. If a relatively slow speed (4.8 kbps) line is used, it will take 360 mili seconds to traverse the composite link if the packets are full; (one half packet time until the next packet starts to be assembled and one packet time till the end of the packet containing the character arrives at the other end and is confirmed as correct. If the packets are not full the time is reduced, though on a slow composite link (4.8 kbps) packets will tend to be full (a single terminal operating at 2400 bps will alone fill half a packet) and errors requiring packet re-transmission will lengthen the time. By the same token if a high speed (eg 80 kbps) composite link is used, the time taken to transmit a full packet reduces to 12 mili seconds and as the majority of packets are less than half full this reduces to 6 mili seconds which is only a six character delay at 9600 bps.

Thus a decision was reached that when slow (less than 15 kbps) composite links were in use (small, distant remote sites), HP specific LSCs running at 4800 bps would be used and where faster (more than 15 kbps) composite links could be justified (all local sites at 19.2 kbs or larger local and remote sites that could justify a BT "kilostream" circuit) normal LSCs running at 9600 bps and achieving a real transfer rate in excess of 7200 bps would be installed. These upgrades have been scheduled for the third quarter of 1983.

With the continued growth of the network (700 terminals projected by 1984) it is reaching the point at which manual surveillance and control procedures would become overwhelmed. To avoid this, plans are in hand to install a BRITOIL designed central network control centre late in 1983. Its detailed design is currently in hand.

10 The Future

In 1985 and 1987 BRITOIL is scheduled to occupy two new custom built office buildings in Aberdeen and Glasgow respectively. Between them, they will hold the majority of BRITOIL staff. Our DCX terminal network has served us well in the past and will continue to do so up until occupancy of the new buildings, but it would be naive to believe that it will suffice for ever. The new buildings give us the opportunity of exploiting Local Area Network (LAN) technology to simplify terminal (and word processor) cabling, increase transmission rates and improve reliability, it would be criminal of us to ignore the opportunity.

11 Acknowledgements

Credit is due to the following BRITOIL staff:

N J Lillie (General Manager) for enunciating our objectives, especially the "single terminal policy" so clearly and forcefully.

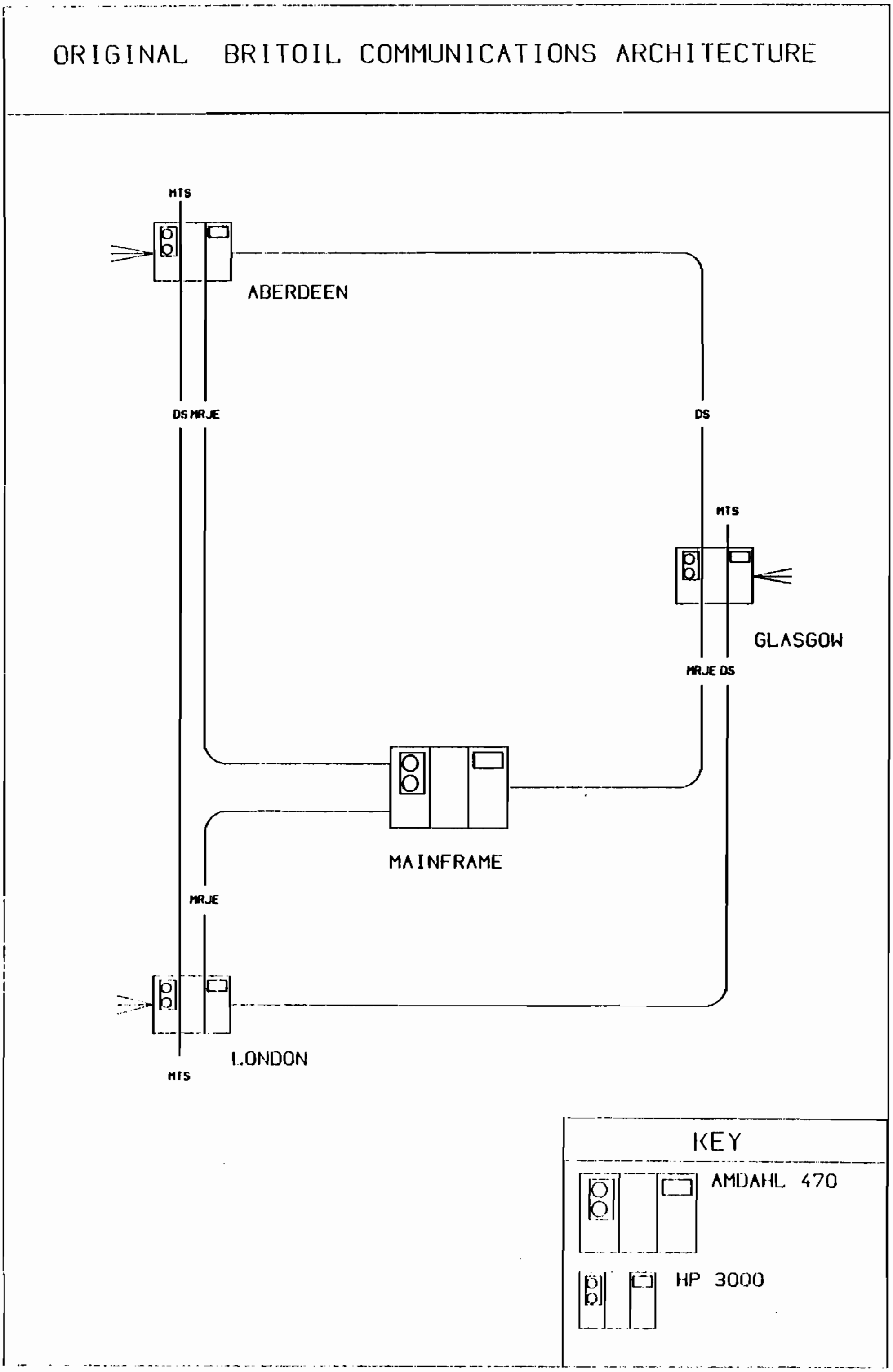
D L Sheldrake (Facilities Manager) for creating the environment in which this work was carried out.

R Mason for defining our initial detailed requirements, and for developing the network through its early and middle stages.

R Andrew for the smooth introduction of the IBM Series/1s.

Credit is also due to : E Piper and I Paterson of CASE for listening to, and understanding, our requirements and steering the development of DCX on behalf of many customers in a way that has satisfied their requirements.

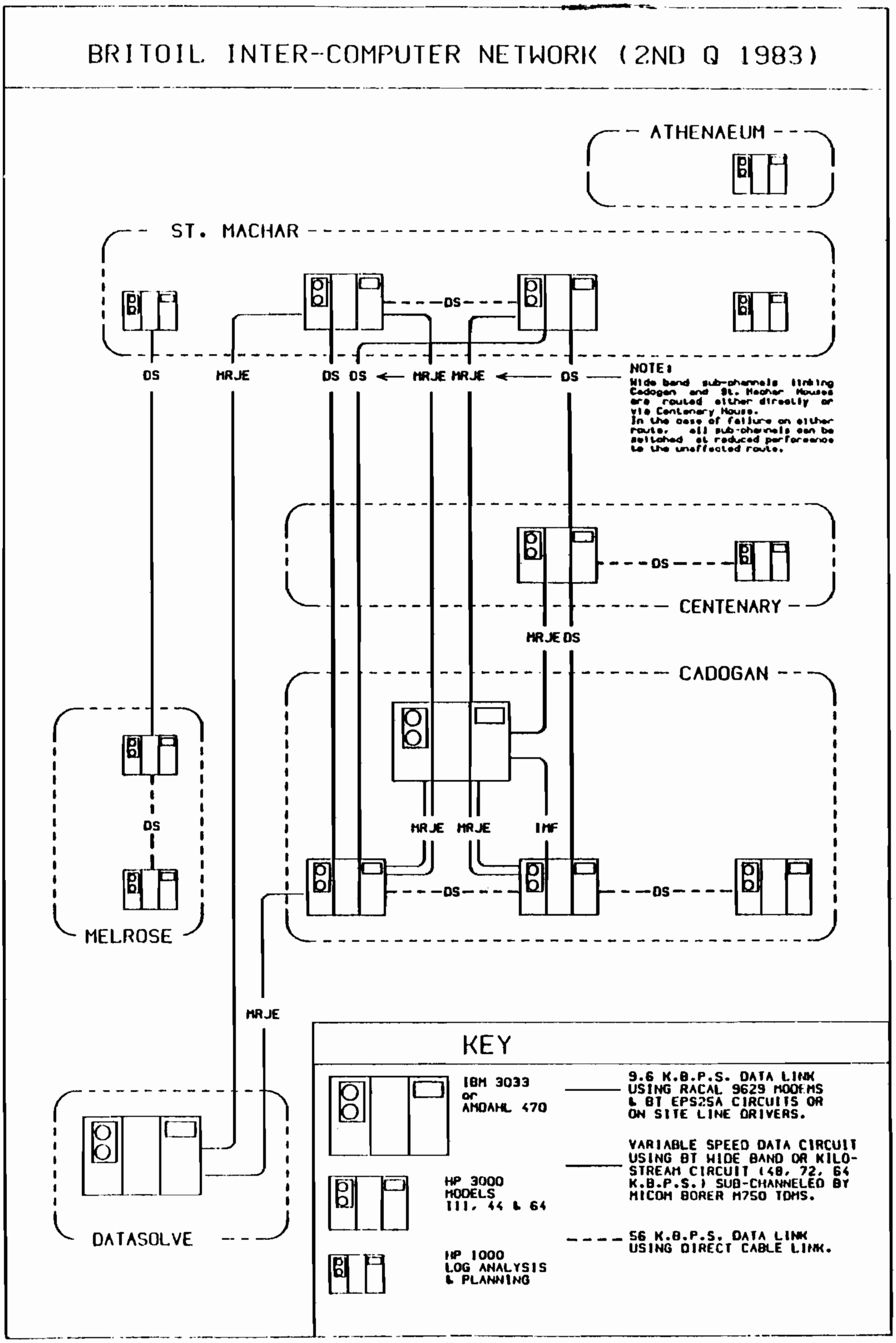
ORIGINAL BRITOIL COMMUNICATIONS ARCHITECTURE



MTS
ABERDEEN
DS MRJE
DS
MTS
GLASGOW
MRJE DS
MAINFRAME
MRJE
LONDON
MTS
KEY
AMDAHL 470
HP 3000

FIGURE 1

BRITOIL INTER-COMPUTER NETWORK (2ND Q 1983)


ATHENAEUM
ST. MACHAR
DS
DS
MRJE
DS
DS
MRJE
MRJE
DS
CENTENARY
DS
MRJE
DS
CADOGAN
MRJE
MRJE
IMF
DS
DS
MELROSE
DS
MRJE
DATASOLVE
KEY
IBM 3033
or
AMDAHL 470
HP 3000
MODELS
III, 44 & 64
HP 1000
LOG ANALYSIS
& PLANNING
9.6 K.B.P.S. DATA LINK USING RACAL 9629 MODEMS & BT EPS25A CIRCUITS OR ON SITE LINE DRIVERS.
VARIABLE SPEED DATA CIRCUIT USING BT WIDE BAND OR KILO-STREAM CIRCUIT (48, 72, 64 K.B.P.S.) SUB-CHANNELED BY MICOM BORER M750 TDMS.
56 K.B.P.S. DATA LINK USING DIRECT CABLE LINK.

NOTE:
Wide band sub-channels linking Cadogan and St. Machar Houses are routed either directly or via Centenary House.
In the case of failure on either route, all sub-channels can be switched at reduced performance to the unaffected route.

FIGURE 2

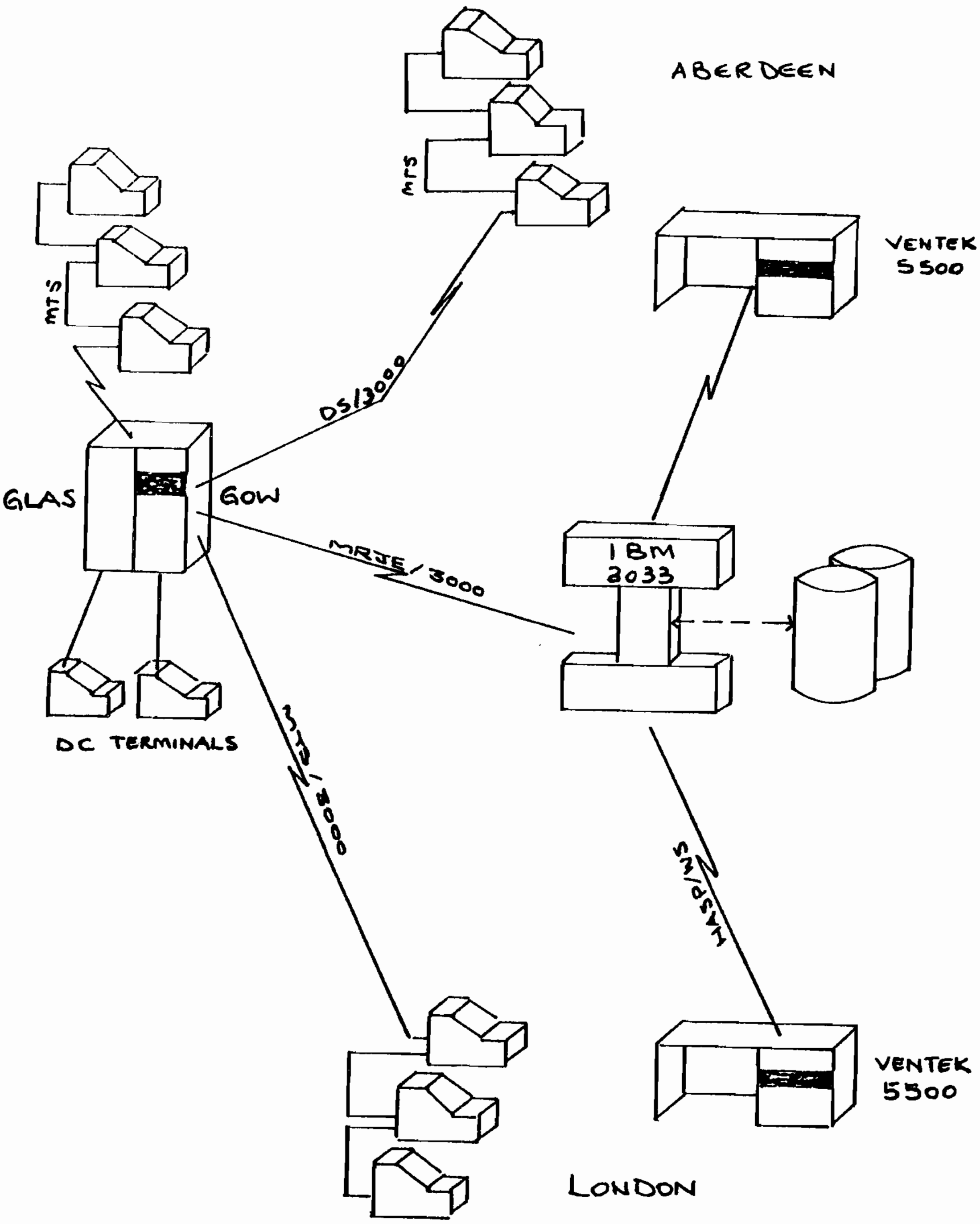

ABERDEEN
MTS
MTS
VENTEK 5500
DS/3000
GLAS
GOW
MRJE/3000
IBM 3033
DC TERMINALS
MTS/3000
HASP/WS
VENTEK 5500
LONDON

FIGURE 3

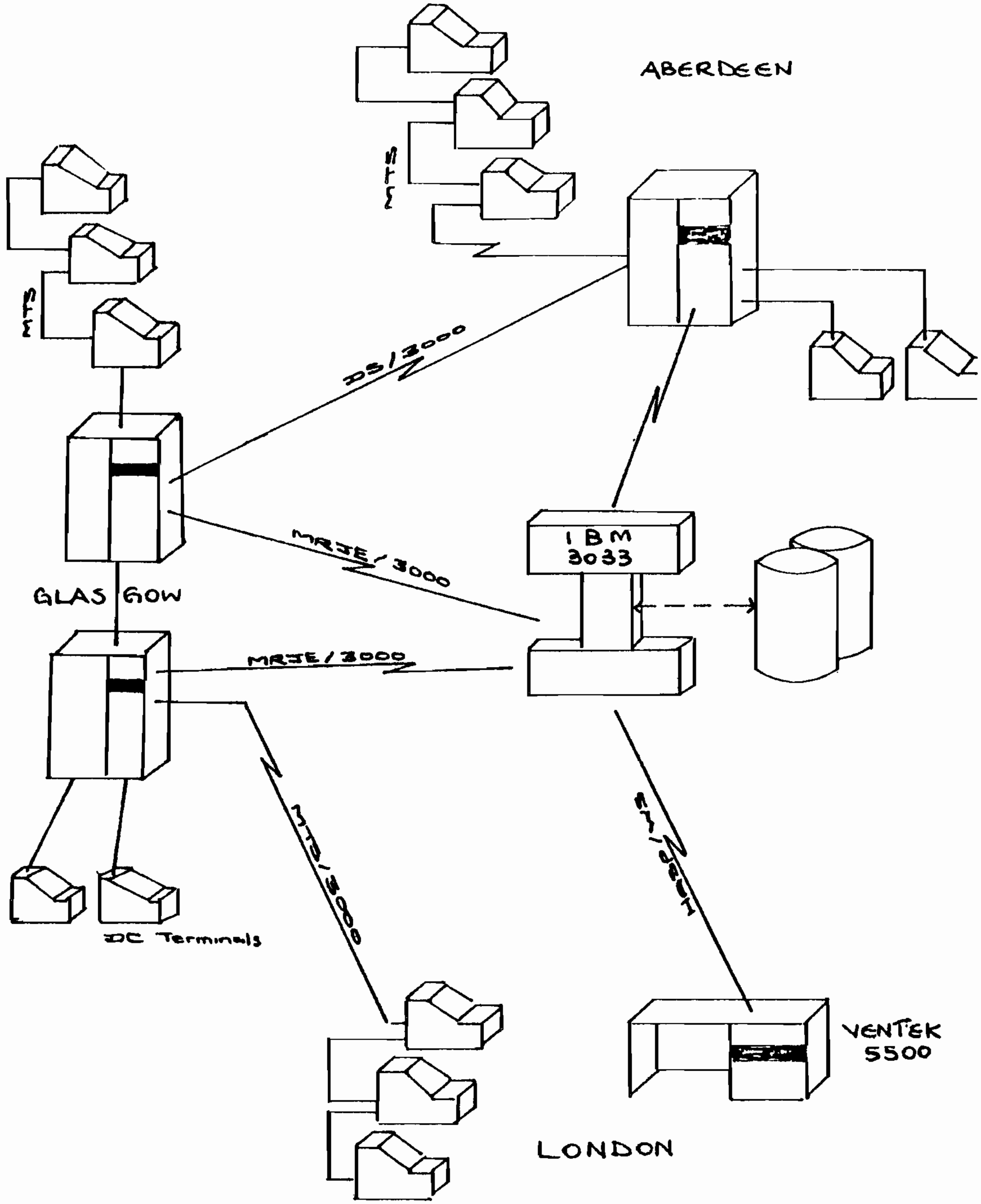
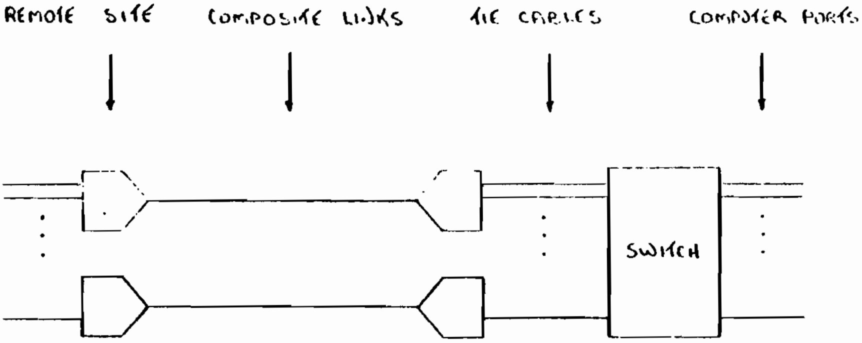
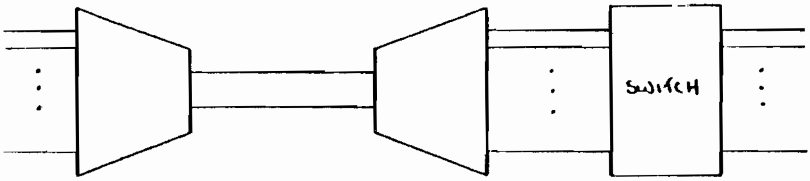

ABERDEEN
MTS
MTS
DS/3000
GLASGOW
MRJE/3000
IBM
3033
MRJE/3000
DC Terminals
MTS/3000
RJE/3033
VENTEK
5500
LONDON

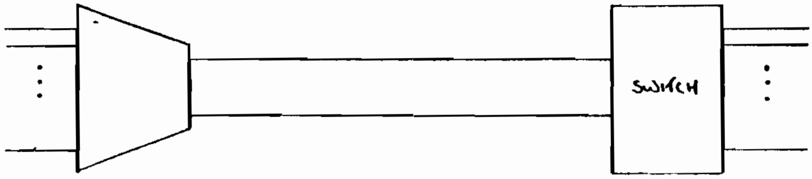
FIGURE 4


REMOTE SITE
COMPOSITE LINKS
TIE CABLES
COMPUTER PORTS
SWITCH

WORST CASE: MULTIPLE MULTIPLEXORS & TIE CABLES TO SWITCH



BETTER CASE: ONE MULTIPLEXOR & TIE CABLES TO SWITCH



BEST CASE: ONE MULTIPLEXOR & NO TIE CABLES

FIGURE 5

3 VENDOR OFFERINGS FOR MULTIPLEXING A LARGE REMOTE SITE TO A CENTRAL SITE WITH A SWITCHING UNIT

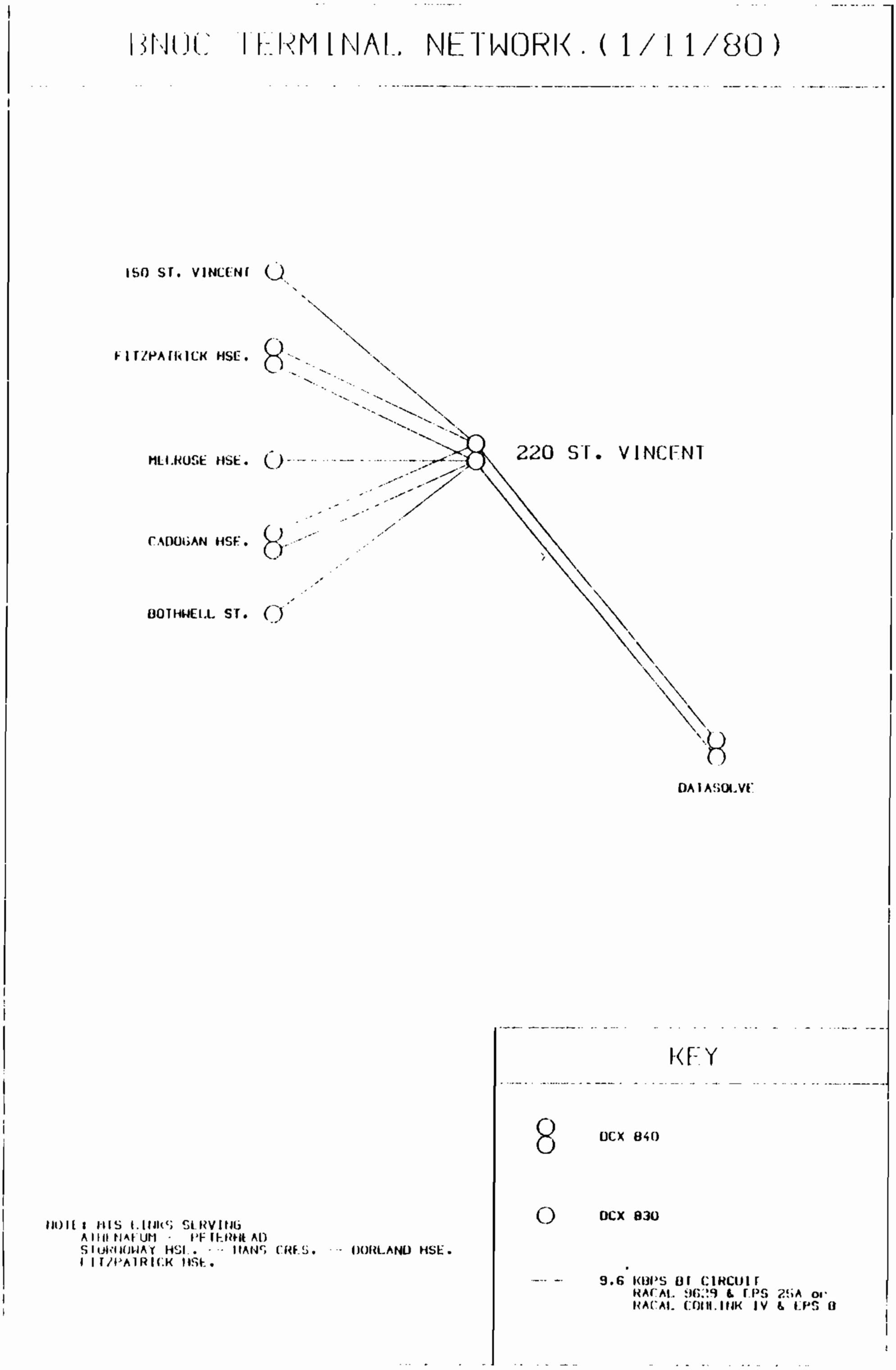
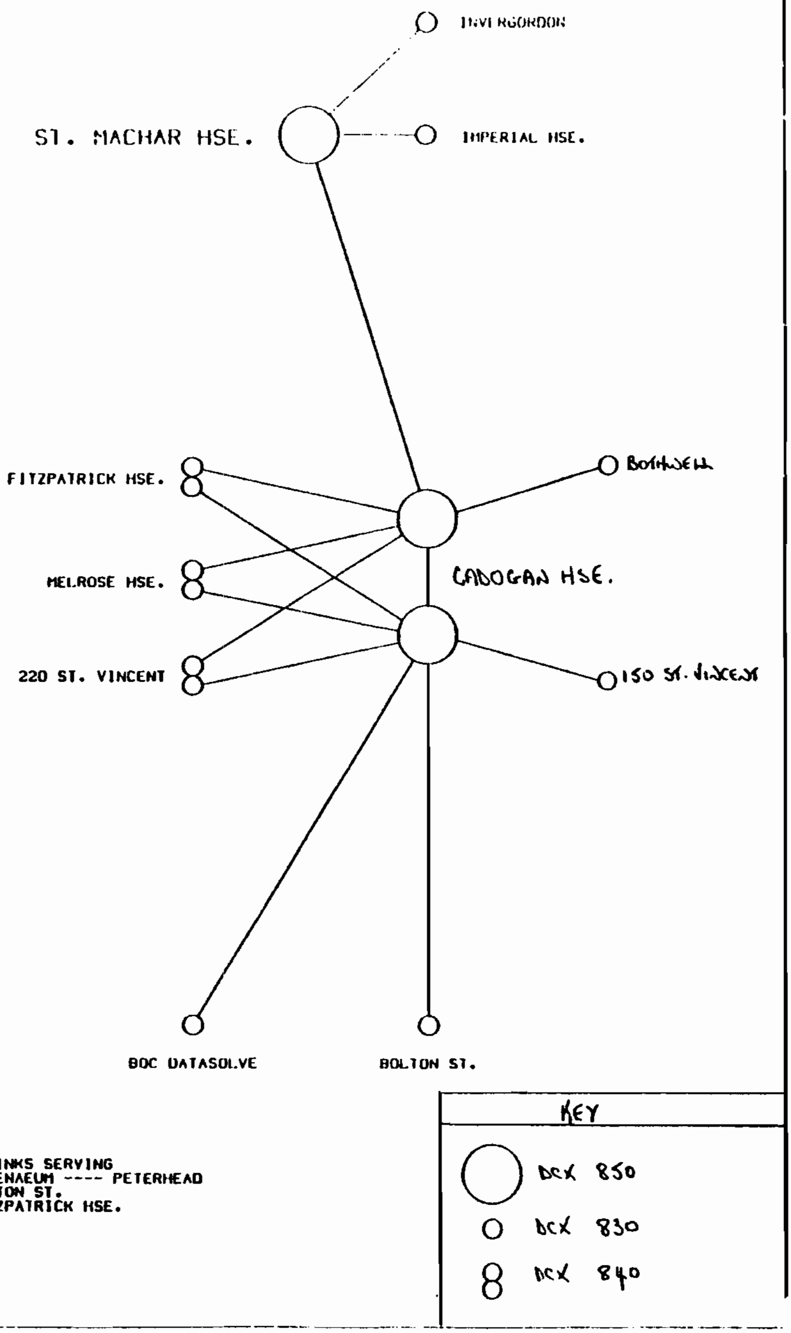

BNOC TERMINAL NETWORK. (1/11/80)
150 ST. VINCENT
FITZPATRICK HSE.
MELROSE HSE.
CADOGAN HSE.
BOTHWELL ST.
220 ST. VINCENT
DATASOLVE
KEY
DCX 840
DCX 830
9.6 KBPS BT CIRCUIT
RACAL 9629 & EPS 25A or
RACAL COMLINK IV & EPS 8
NOTE: BTS LINKS SERVING
ATHENAEUM - PETERHEAD
STORNOWAY HSE. -- HANS CRES. -- DORLAND HSE.
FITZPATRICK HSE.

FIGURE 6

BNOC TERMINAL NETWORK (1/1/82)


INVERGORDON
ST. MACHAR HSE.
IMPERIAL HSE.
FITZPATRICK HSE.
BOTHWELL
MELROSE HSE.
CADOGAN HSE.
220 ST. VINCENT
150 ST. VINCENT
BOC DATASOLVE
BOLTON ST.
KEY
DCX 850
DCX 830
DCX 840

NOTE: DIRECT MTS LINKS SERVING
OPS ---- ATHENAEUM ---- PETERHEAD
PROD --- BOLTON ST.
PROD --- FITZPATRICK HSE.

FIGURE 7

BRITOIL TERMINAL NETWORK (2ND Q 1983)

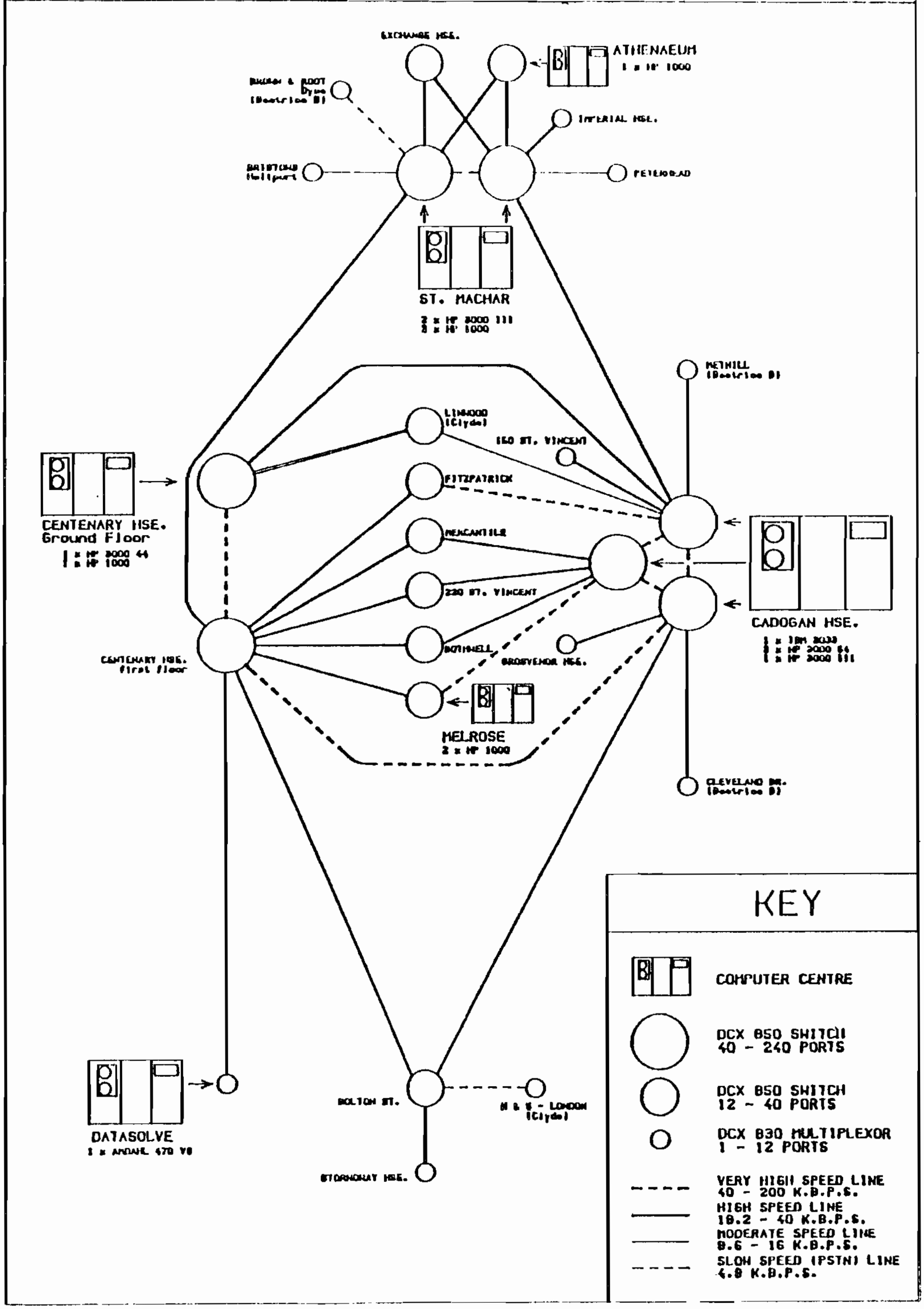

EXCHANGE HSE.
ATHENAEUM
1 x HP 1000
BROWN & ROOT
Dyce
(Beatrice B)
IMPERIAL HSE.
BRISTOW
Heliport
PETERHEAD
ST. MACHAR
2 x HP 3000 III
3 x HP 1000
NETHILL
(Beatrice B)
LINWOOD
(Clyde)
150 ST. VINCENT
FITZPATRICK
MERCANTILE
230 ST. VINCENT
BOTHWELL
GROSVENOR HSE.
MELROSE
2 x HP 1000
CENTENARY HSE.
Ground Floor
1 x HP 3000 44
1 x HP 1000
CENTENARY HSE.
First floor
CADOGAN HSE.
1 x IBM 3033
3 x HP 3000 64
1 x HP 3000 III
CLEVELAND DR.
(Beatrice B)
DATASOLVE
1 x AMDAHL 470 V8
BOLTON ST.
N & S - LONDON
(Clyde)
STORNOWAY HSE.
KEY
COMPUTER CENTRE
DCX 850 SWITCH
40 - 240 PORTS
DCX 850 SWITCH
12 - 40 PORTS
DCX 830 MULTIPLEXOR
1 - 12 PORTS
VERY HIGH SPEED LINE
40 - 200 K.B.P.S.
HIGH SPEED LINE
19.2 - 40 K.B.P.S.
MODERATE SPEED LINE
9.6 - 16 K.B.P.S.
SLOW SPEED (PSTN) LINE
4.8 K.B.P.S.

FIGURE 8

'Interprocess Communication in a Network Environment'

by

Nick Bates and Gary Wolstenholme (Hewlett-Packard UK Ltd)

A) INTRODUCTION

Since the introduction of the HP3000 Distributed Systems Product, DS/3000, in 1977, many customers have implemented business systems involving application programs executing on one computer system which require access to data structures held on others.

MPE, together with its data management and data comunication subsystems, provides many (arguably too many) software facilities by which application designers and programmers can implement systems which require 'remote data access'.

One possibility allows local application programs the ability to directly access remote files using Remote File Access (RFA) or Remote Data Base Access (RDBA).

Alternative software facilities exist to allow inter-process communication. This enables conversations to take place between local and remote applications programs using Program-to-Program Communication (PTOPC) or the Interprocess Communication (IPC) 'message files' provided by the MPE File System.

The current DS/3000 product supports each of the above faciiities between physically connected nodes in a point-to-point DS network, using switched/leased/hardwired connections, and between logically connected nodes in a true data communications network, for example, an X.25 Public Data Network.

Given, a straightforward 2-node arrangement, 'remote data access' is functionally simple to achieve using any of the above techniques. However, when an application designer/programmer is presented with a complex topology of HP3000 computer systems, the mechanism of providing generalised 'remote data access' is by no means obvious. This situation is further compounded by the fact that, typically, given the same problem, no two designers/programmers would arrive at the same solution.

Whichever techniques are chosen to facilitate remote data access, the ultimate design should be applicable to networks, no matter how simple or complex, and should be characterised by (in no particular order of importance)

- Ease of development and testing

- Minimal use of system table resources

- Acceptable performance and ease of monitoring same

- Minimal disruption in the event of a communication failure with associated ease of recovery

- Ease of operator control in terms of startup, shutdown and intervention.

- Resilience to change in network topology

This paper is concerned with evaluating the advantages and drawbacks of using the various basic tools in developing interprocess communications and suggests a skeleton design for a software layer to form the basis of a generalised 'message transfer facility' which may be used by applications to perform 'remote data access'.

B) THE ALTERNATIVES.

The alternative basic tools available to implement a network on HP3000s can be divided into two major groups.

(a) Virtual Terminal.

The simplist technique available to access data on a remote HP3000 is to use the Virtual Terminal facility within DS/3000. The user issues a DSLINE command to gain access to the communications line and then 'logs on' to the remote HP3000 as if he is 'hardwired'. The security for the session running on the remote machine is established in the normal manner using the remote logon id. The user now has access to the remote system and can 'flip-flop' between the local and remote machine. The full set of commands and subsystems are available to the user both locally and remotely, however in this mode only one program can be run on the users behalf at any given point in time.

(b) Remote Data Access.

An extension to the virtual terminal approach outlined above, is for a program at one end of the communication link to directly access a data file or data base residing on the other HP3000. There are various methods of implementing this type of remote data access approach to an HP3000 network.

(1) Remote File /Data Base Access (RFA/RDBA)

Once a user has established a remote session, he can access files and data bases by using the DEV= parameter of the MPE FILE command. This facility requires no special programming, as DS/3000 handles all the communications to access the remote data. The program is written without necessarilly considering the location of a particular file

(2) Program To Program Communication.

A local program can initiate a 'slave' program on a remote HP3000 once a remote session has been established. This is achieved using a set of Intrinsic calls which will also facilitate the transfer of data and parameters between the two processes. The normal operatition of such a system is for the master to transmit a message to the slave containing data and/or parameters. The slave program then performs the requested function and returns data and/or parameters back to the master program. Only one of the master/slave processes is active at any one time. Both the master and slave programs are relatively easy to code but care must be taken with the logic of such a system.

(3) Inter Process Communication or Message Files.

IPC files offer a facility for one or more users to send a message via the file system to another user, without the family tree contraints of the MAIL Intrinsics. IPC files can be used as an alternative method to PTOPC in a network environment, offering the same functions as PTOPC but removing the constraint of only one process being executable at any given time. Once a write to an IPC/Message file has been completed, the process issueing the write can continue execution without awaiting a response.

C) DESIGN CONSIDERATIONS.

There are several factors which need cosideration when deciding upon a method of implementing a network on HP3000s and we should consider the pros and cons of the different basic tools available to us.

a) Virtual Terminal

This method offers a solution that is both easy to develop and simple to test, as the remote program is written as if it is being used by local users. The main limitation of this technique is an operational one, in that it requires the user to exit the local program and then run the remote program to access the required data. Once the data has been obtained, the user must then exit the remote program and rerun the local one. This is also poor from performance considerations. It requires repetative loading and running of both local and remote programs, and also uses up system resources. For each user accessing the remote HP3000, there is additional CI and user processes, using up both memory for code and data, and also CST,DST,PCB and DRT entries. The responsibility for recovery from a communications failure when using this approach, is with the individual user.

b) RFA/RDBA

This method again offers ease of developmant and testing as it can also be performed using only the local HP3000. It removes the user intervention requirement of the virtual terminal approach as the program can directly access the remote data. The two main drawbacks of RFA/RDBA are in resource utilisation and performance. Here again each user requires remote memory and table entries. The bottleneck of most networks is the thoughput of the communications link itself, and to ease this and hence improve overall responses, the system designer should minimise the amount of traffic on the line for any individual program. The RFA/RDBA approach is often very inefficient in its use of the communications line, as it usually requires several reads/writes to a file/data base to obtain the requested information.This produces a large amount of unnecessary traffic on the line and hence impinges on the performance of the communication system as a whole. In the event of a failure in the communcations link, each RFA/RDBA program must handle its own recovery.

c) PTOPC

A program to program communications technique can make more efficient usage of the communications link, as logic can be written into the master/slave programs to transmit only that data which is essential to the users request. The logic of such a system is more complex, and the nature of PTOPC requires two HP3000s to both develop and test a master/slave link. If one master/slave link is used per user, the resource utilisation and line recovery problems are simmilar to those in an RFA/RDBA environment. If we choose to try and use one master/slave pair as a transport mechanism for several users we must be aware of the severe performance implications of such a system. The nature of PTOPC restricts us to servicing one request at a time. While this request is being serviced for one user, all other users requiring to use the transport mechanism are waited until the current request has been serviced. If this request involves searches down long chains of a data set, the overall response from the line is severely degraded. Whilst using PTOPC as a general transport mechanism requires less resources on the HP3000, and also means that only one process needs to handle line recovery/failure, the performance implications normally far outway these advantages.

d) IPC Files

IPC files offer all the advantages of a PTOPC approach but have some added bonuses. The main difference with using IPC files as a transport mechanism for multiple users is that once the control process has written it's message to the remote IPC file, it is not waited upon a response. In this way several requests can be serviced 'simultaneously', without awaiting a response to one service request before transmitting the next. Unlike PTOPC, testing and development of an IPC file system can be performed on a single HP3000, by writing to a local IPC file during this phase. IPC files offer increased line efficiency over PTOPC in a single transport mechanism environment, while offering the advantages of control and recoverability of a single process transport mechanism.

D) SUGGESTED DESIGN FOR 'GENERALISED MESSAGE TRANSFER'

a) Terms of reference

The flow of data for the suggested design of a generalised 'message transfer' facility is illustrated in Fig.1.

This structure caters for (and is limited to!) the transmission of 'remote data access' requests from local 'application' processes to remote 'server' processes, and the transmission of the replies returned to the applications from the remote 'server'.

In this context, 'application process' implies an interactive or batch process, handling transactions input from terminals or files respectively. The term 'server process' implies one whose sole function in life is to perform access to locally held data structures, based upon the contents of remotely generated request messages, and to return an appropriate reply message.

Fig.1 shows the data flow between a local 'application' process (A) and a remote 'server' process (S) in a simple 2-node environment. This is done for reasons of clarity - the design caters for simultaneous bi-directional message transfer between multiple 'applications' and multiple 'servers' potentially on differing remotely connected systems.

In addition, although handled at the 'application level' rather than the 'DS subsystem level', this design is applicable to environments where 'remote data access' needs to take place via an intermediate node (or nodes).

b) Major components

Clearly, some key players in Fig.1 need identifying before any discussion of data flow can take place. These key players are the controller process (CONTROL) and transmitter process (TRANSMIT) on each system.

CONTROL is responsible for the routing of both incoming and outgoing 'application' requests and 'server' replies. Thus, each message should contain control information about the sender, receiver and the source and destination system identifiers. In the suggested design, each system requires one CONTROL process.

TRANSMIT is responsible for the physical transmission of outgoing 'application' requests and 'server' replies across a specific communications link via DS/3000. In the suggested design, there are two TRANSMIT processes for any communication link, one on each side of the link.

One aspect should be evident from Fig.1. If this structure is to succeed, the CONTROL process needs to be in complete command of the situation. Consequently, CONTROL needs to know which 'application' 'server' and 'TRANSMIT' processes are participating in the game, and also requires timely information on communication line errors. Thus, any implementation of this basic design should ensure that all processes 'sign up' to participate in the message transfer facility by placing a 'request to sign up' in the input message file, MSG, used by the CONTROL process. Simililarly, orderly termination of processes is desirable, and all participants should 'request to sign off' in a similar fashion. CONTROL would typically hold tables in his stack reflecting the current state of the environment. If each participant adheres to these conventions, no delinquent processes become involved and no process terminations are missed.

All communication files shown in Fig.1 are Interprocess Communication (IPC) message files provided by the MPE File System.

c) Data flow - requests and replies

Assuming that all processes in Fig.1 have 'signed up' to the message transfer facility and all channels of communication via IPC files are open and quiescent, we can now examine the data flow of an 'application' request to the 'server' and the return of the servers reply.

1) Local application process (A) writes a request to the local common message file, MSG, and posts a read on its private message file, AMSG, to wait on the servers reply.

2) Local CONTROL process reads the request from MSG and performs some basic parameter checking on its content. For example, does the destination system identifier appear in its tables as one to which a local TRANSMIT process has a current access path.

3) Local CONTROL process having verified the request parameters (as far as is possible at this stage) writes the request to the appropriate OUTPUT file for transmission.

4) Local TRANSMIT process reads the request from OUTPUT and writes it to the remote INPUT file using RFA in step 5.

6) Remote CONTROL process reads the request from INPUT and performs basic parameter checking. For example, does the required server process appear in its tables as one which is currently active.

7) Remote CONTROL process having further verified the request parameters writes the request to the private message file, SMSG, of the 'server' process (S).

8) Remote server process (S) reads the request from SMSG.

9) Remote server process (S) performs whatever accessing of remote data structures is required by the request (at last!) and formats its reply.

10) Remote server process (S) writes its reply to the remote common message file (MSG) and posts a read on its private message file, AMSG, to await the next request.

11) Remote CONTROL process reads the reply from MSG

12) Remote CONTROL process writes the reply to the appropriate remote OUTPUT file.

13) Remote TRANSMIT process reads the reply from the remote OUTPUT file and writes it to the local INPUT file using RFA in step 14.

15) Local CONTROL process reads the reply from the local INPUT file

16) Local CONTROL process writes the reply to the private message file, AMSG, of the originating application process (A).

17) Local application process (A) reads the reply from AMSG and continues execution.

d) Benefits of a generalised 'message transfer system'

Clearly, the implementation of the above design requires a significant amount of foresight and programming effort to turn concept into reality. At first glance this does not appear such an attractive proposition, especially since alternatives such as direct Remote Database Access calls are functionally simple to code into individual programs.

However, anyone contemplating implementing a network of HP3000 systems requiring 'remote data access' in transactions, would be well advised (by those qualified through the school of hard knocks) to seriously investigate the above skeleton design. Carefully implemented, the end result should more than justify the initial investment.

Let us examine the salient aspects of this design from the standpoints and criteria outlined in the Introduction.

*) Ease of development and testing

Applications programmers should concentrate on data access and data processing! The generalised structure above isolates the application programmer from DS/3000 entirely. Notably, the only program involved in DS/3000 remote logons, logoffs and RFA is the TRANSMIT program. Consequently, this is the only one required to interpret and handle exceptional error conditions returned by DS.

The 'message file' characteristic of an IPC file is a function of the manner in which the file is built, and not of the code which accesses it (for simple reads and writes). Whilst it is true that language compilers such as COBOL and FORTRAN (and pseudo-compilers such as TRANSACT) generate code which calls the MPE file system intrinsics based upon language statements, it is recommended that applications programmers are provided with a series of high-level procedures. These would most likely be written in SPL and able to perform any extended use of 'message files' which may be required such as timed-reads, nowait I/O etc.

One advantage of the above design is that applications can be tested out on a single system by merely having the TRANSMIT process write to a local file instead of a remote one. This, incidentally is the basic disadvantage of PTOPC, which requires 2 systems to perform testing of a master/slave pair (preferably located in the same room!).

The above design only involves one mandatory use of an HP3000 special capability. This occurs within the CONTROL program whose quiescent state is to be suspended awaiting the arrival of an input message in either the MSG file or any of the INPUT files. This is accomplished by having CONTROL open the MSG and INPUT files as nowait I/O files for read access. Thus CONTROL would post FREADs against each of its input files and wait on an IOWAIT(0). This construct awakes CONTROL whenever an input message arrives in one of his input files, and identifies the file concerned.

*) Minimal use of system table resources

It is one thing for a single application program to perform its own direct remote file accessing using RFA/RDBA, and another for a whole raft of these programs to simultaneously perform such functions. Contrary to popular opinion, any given HP3000 configuration does have a finite limitation on the number of remote sessions which can be simultaneously handled, as a result of the limitation on the number of IODSTRM0 devices which may be configured.

The above design requires only only 2 remote sessions to be initiated for any given communication link (one on each side of the link).

The designer should be aware, in this design as in any other, of the resource consumption of his software. Limitations still exist on, for example, the number of files which a single process may simultaneously have open, and on the number of available data segments (DSTs) which may be used for MPE file buffering.

*) Acceptable performance and ease of monitoring

Perceived response times of transactions which access remote computers are largely limited by the line speed and contention for the line. The latter is largely determined by the number of logical accesses to a line by simultaneous application programs. Using the above design an 'application' request may say 'chain through a remote orders detail data set and return sales figures for a specific product'. Using the generalised 'message transfer facility' this would involve two logical accesses to the line, one for the request and one for the reply. Using individual Remote Database Access calls this would require an indeterminate number of logical acesses using remote DBGET calls, with the corresponding impact on the elapsed time for the transaction, and performance in general.

The generalised 'message transfer facility', at first glance, appears to involve an inordinate number of IPC files (disc files). In reality, it is likely that a number of interprocess communications would merely involve memory-to-memory transfers, from the writers stack, through the file system buffers, to the readers stack. This would be the case if the CONTROL process was to run at a high enough priority to enable its reading of the MSG file to keep pace at which the various 'application' and 'server' processes perform writes to same.

Computer Museum

Performance monitoring of the system CPU, memory and disc I/O resources consumed by a process (or processes) is easily achieved by the existing HP3000 Performance Tools (OPT and MPEDCP). These programs, do not, unfortunately, volunteer information on the utilisation of a communication line on a program-by-program basis. The design of the generalised 'message transfer facility' was done with this in mind. All data transfers, from application to server processes, and vice versa, necessarily involve the CONTROL process which is therefore in a position to monitor and volunteer information on message traffic by individual participants.

*) Minimal disruption in the event of a communication failure, and ease of recovery

In the event of a line failure on any given communications line, only two processes suffer in this structure (the TRANSMIT processes). Since the generalised 'message transfer facility' design dissociates 'application' and 'server' processes from the data communication aspects, all transaction messages are buffered in the OUTPUT files relating to the communication links. These transactions could be routed to their destination processes, on restarting the TRANSMIT processes.

*) Ease of operator control in terms of startup, shutdown and intervention

The HP3000 Console Operator commands =LOGOFF and =SHUTDOWN do not provide the orderly termination procedures required to ensure logcal consistency of application databases and programs, required in a distributed data processing environment. Implementation of the above design should provide the ability for a central operator to indicate imminent system shutdown to the processes participating in the message transfer environent. This would be most sensibly performed by providing a direct interface for an operator program to the MSG file, thus keeping the CONTROL process informed of this and other major system events.

*) Resilience to change in network topology

The addition of a new node into an X.25 network would necessitate little change to application programs, however their 'remote data access' is performed. The majority of HP3000 nodes in a DS environment are connected in a point-to-point fashion and the introduction of an additional node can have a significant effect on the manner in which 'remote data access' is performed. For example,

```
Yesterday : (node A)--------------------(node B)

Today     : (node A)------(node C)------(node B)
```

Anybody for leapfrog?

Whilst it is possible that future implementations of DS/3000 may provide pass-through and nodal addressing (in a similar manner to DS/1000) the current product does not permit direct access to data structures on anything other than a physically connected system in a point-to-point configuration. Consequently, re-routing of messages needs to be handled at the 'application level'. In the case of the generalised 'message transfer facility' this could be performed by the CONTROL process. Such re-routing would require information about the network configuration to be resident on each system.

E) SUMMARY ----------

We have oulined the facilities available to the systems designer when choosing a technique for a communications network and suggested a design for a generalised message transfer system. This design offers

- Ease of development and testing
- Minimal use of system table resources
- Acceptable performance and ease of monitoring same
- Minimal disruption in the event of a communication failure with associated ease of recovery
- Ease of operator control in terms of startup, shutdown and intervention.
- Resilience to change in network topology

by utilising the software currently available on the HP3000.

This design has not been implemented onto a HP3000 network and we would strongly suggest that anyone contemplating such a network should seek competent advice before embarking on implementation.

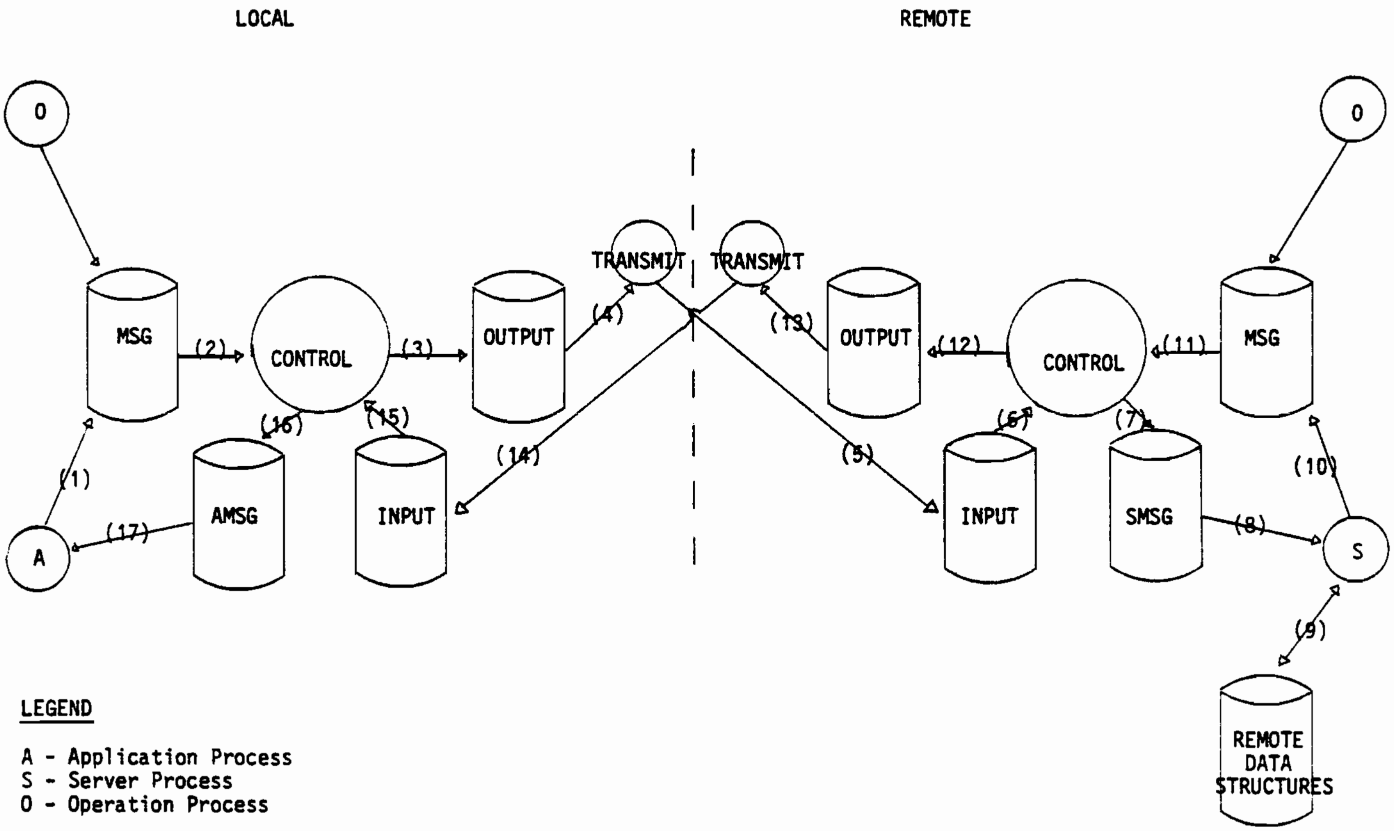

LOCAL
REMOTE
O
MSG
(2)
CONTROL
(3)
OUTPUT
(4)
TRANSMIT
TRANSMIT
(13)
OUTPUT
(12)
CONTROL
(11)
MSG
O
(1)
A
(17)
AMSG
(16)
(15)
INPUT
(14)
(5)
INPUT
(6)
(7)
SMSG
(8)
S
(10)
(9)
REMOTE DATA STRUCTURES
LEGEND
A - Application Process
S - Server Process
O - Operation Process

FIG 1 DATA FLOW FOR GENERALISED MESSAGE TRANSFER

Anticipating OSI

by

Bjørn Vermo

Fjerndata, Norway

The standards situation

Anybody working in EDP the last years will have heard how much simpler everything will be when all equipment and software works according to **OSI** specifications. Then all computers, terminals and software from all vendors will work together in perfect harmony, and everybody will be happy.
Alas, we have still quite a way to go! The framework can be found in ISO/DIS 7498, a document of some 100 pages describing the basic philosophy of Open Systems Interconnection. When it comes to the actual details, however, the necessary amount of detail is considerably greater. Also, the fact that these standards must be made and agreed by international committees do not exactly hasten things along. Still, much progress has been made, and today the standards up through the transport layer are sufficiently refined for manufacturers to release implementations. The interest in Local Area Networks is helping this process along.

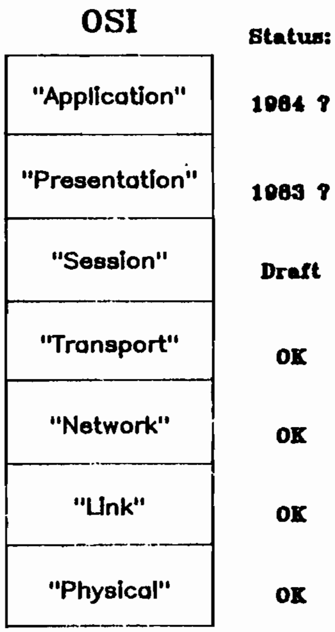
The development of user needs

When the first interactive applications arrived, users were happy with a crummy terminal which could access one application on one computer. True, they would grumble a bit about features of the application and details in the terminal dialogue, but by and large they found on-line applications to be so much more productive than batch processing that the interest of everybody was focussed on that. As soon as the second application was on the air, however, most organizations discovered that there were people who wanted access to "unrelated" applications from the same terminal. Not all computers were set up to handle this. Minicomputer users usually managed quite well, but many applications running on the larger mainframes were incredibly inflexible.
Of course, the next user demand was to use the same terminal to access different computers from different manufacturers - preferably at high speed and with short response times. Very slowly multiprotocol terminals, black-box protocol converters and emulator software appeared. The problem for the poor user is that these solutions usually are anything but easy to use.
Mainframe software has also improved so that it is possible to switch from one application to another. However, anybody who has had occation to use IMS, TSO and VM/CMS from the same 3270-type terminal will deny that it is easy to switch, given the number of different panels you have to go through.
User needs are, however, increasing at a faster pace than solutions. Electronic mail is useful, but only if you can communicate with all the people you usually need to reach. So how does the Singapore office send messages to the Rio office? Probably by telex, because it is already there.
In most organizations, somebody will need information from one of the many data banks which have been established. These are usually connected to public packet-switched networks, and expect the user to log on from an asyncroneous terminal connected to a X.28 PAD. Nice and standardized, is it not? So how many users are actually accessing their ordinary day to day applications in this way? Even if they use an asyncroneous terminal and you supply a modem switch, how many know how to change the speed and maybe parity settings? And even if they know how to, they have to log off from their ordinary system, switch modems, call the PAD, log on to the network, log on to the data bank system and in some cases even to the application they want to use there. Hardly what I would call easy and efficient.

What happens in real life, is that a dedicated terminal is installed. And the users who only need this kind of information now and then have to ask somebody else to access it for them, since it is not cost-efficient to provide them with terminals of their own.

What we will get - sometime

When, someday, all software (maybe) has been recycled according to OSI specifications, everything will communicate efficiently with everything else. Peer-level protocols will ensure that maximum use is made of the capabilities of the communicating entities so as to make life as pleasant as possible for the user. Knowing something about software development, I suspect that this blissful situation may yet be a little while into the future. Most existing systems are quite "closed", and as long as even the basic standards are unfinished there is no reason to expect any rush of hardware, firmware or software. The development in the lower layers of the model has been encouraging, but pure communication has not been as deeply integrated into applications as file handling and formatting or display control has. Hence, presentation and application layer implementations will come more slowly than X.21 and X.25 did. Protocol conversion in the form of "layer 6 gateways" will be with us for a long time to come.



OSI	Status:
"Application"	1984 ?
"Presentation"	1983 ?
"Session"	Draft
"Transport"	OK
"Network"	OK
"Link"	OK
"Physical"	OK

What we can provide

Since Fjerndata is a major IBM service bureau with hundreds of batch terminals and a large and rapidly increasing network of IBM 3270-type terminals, our main interest in mang "openings" between closed systems has been in order to provide access to IBM-bound applications from non-IBM computers and terminals.
HP3000 has proven to be more suitable for this purpose than other computers we have been working with, and has been the target for our major effort. This is VTS/IMAS, which started out as a way to provide IBM 3270-access from asyncroneous terminals operating in character mode. As new features were added, it developed into a more powerful system with capabilities not found in IBM terminals. The most important of these is the provision for **automatic dialogues**, making it possible to automate routine operations like signing on or off, copying screen images to local files and vice versa. These capabilities soon encouraged experimentation, and it proved possible to make a primitive gateway between HPMAIL and our IBM-based message switching system. This has in its turn caused improvements in the auto dialogue language to make it more flexible. Other features, such as the capability to run several applications at the same time, increase the usefulness of this software. A user can connect to all of his applications just by executing a UDC, and can switch from one to the other by the push of a function key. The applications can be on different computers, both HP, IBM and IBM look-alike. Presently, only HP terminals are supported, and since presentation level protocols are not standardized our software is of necessity not "open" in the OSI-meaning. However, we have done things in such a way that when the standards are

approved it will be fairly straightforward accomodate them. The structure is already defined in ISO 7498, and it is possible to know a lot about what is going to happen from the working documents.

We believe we have a good starting point for further development, and hope to provide more extensive virtual terminal services in the future. Access to HP from IBM is one interesting possibility, use of non-HP terminals in applications requiring format mode another, and networking with personal computers is definitely interesting.

Important do's and don't's

Even if your software is not communications oriented, it may be wise to prepare for a future of open systems. The lack of capacity to make systems even though they are obviously profitable will be even more of a problem in the future, user-tools nonwithstanding, and there is a high risk that systems will be used longer and for more things than the original designers intended. Therefore, it is wise to make everything as open-ended as possible. The first thing you must do in order to anticipate OSI in your current software developments, is to stick to current standards as far as possible. To do this, you must know what the standards are. Failing that, you end up like HP, IBM or many others, by making something that is almost standard or commonly used - meaning it will not work together with anything else unless the sun is shining and north-easterly winds prevail.

In order to know the standards, it is useful to subscribe to them from your own national standards organization, which will sell subscriptions to international standards in the area of your interest. It is also a good move to find out the intentions behind the standards. If not, it is very easy to misinterprete things. Look at HP display terminals. In a seemingly very intelligent move, they have made terminals fairly international by implementing the "extended roman" character set. There is a little catch. The way it is implemented, it can only work in one language at a time. By reading the definition of the ISO 7-bit character set with 8-bit extensions first, all problems would have been avoided.

It is easy to find other examples. Both communications and international use is particularly unforgiving of sloppy system specification and implementation.

Probably, most system designers will feel like forgetting the whole thing after reading OSI and a couple of ISO working documents. Take courage! In all probability, most of the possibilities will not be of any concern for your application. The important thing is to make a well-structured system, modularized according to the OSI 7-layer model. Do not worry too much about the fact that the system software you interface into does not have a service interface anything like what is specified. The important thing is to separate different functions and gather similar functions so that it will be easy to modify and adapt later on. Write application layer software pretending that everything from presentation and down is available and functioning. Then you write an interface to provide the actual interface to the system. Do not specify your display format inside the application - provide for some variation. Remember that constants do not exist in reality - use a parameter file instead. The same goes for error messages. Even if you are sure that your software will never be used in another country or by any kind of minority group, it is still much easier to maintain a message file. And surely everybody realize thaheadings and such are obviously data and thus have no business being on a program file? The partition between layer 6 - presentation - and layer 7 - application - is useful even when you operate inside a single system. Actually, the whole OSI-model is good design even if you want to make a closed system which is not supposed to communicate with anything. The separation of different functions will make everything easier to maintain and much mor[illegible]ice-independent.

Benefits

If your applications are made with a standardized interface, it will be easy to run them interactively from other programs. In the near future, it will be possible to make systems where each statement will execute an application. With ever more powerful computers, this will be one way of bringing that power to the user workstation. It may be the only way to shorten the development cycle sufficiently in many cases, even though "automatic programming" and user-definable systems will probably improve and be more important.

What the user will get

As we steadily move towards our goal of open systems and the computer networks spread out across the world, our user will get the opportunity to be much more productive. Running a spreadsheet on his desktop micro, he will gather data from many different computers without ever knowing about it, and with the system quietly checking in the background that norhing illegal is taking place. Exchange rates and international trade figures will be available - in a format suitable for the application at hand, and messages will arrive from around the world. Should it be necessary to travel somewhere, the system will know where to get flight information. It will even handle the booking. And - it will even be possible to get an answer out of the corporate IMS database on the IBM!

THE STATE OF AFFAIRS WITH HEWLETT PACKARD'S X.25 PRODUCT OFFERING

HEWLETT PACKARD CO.
GRENOBLE NETWORKS DIVISION

HP 3000 USERS GROUP
OCTOBER 1983

THE STATE OF AFFAIRS WITH HEWLETT PACKARD'S X.25 PRODUCT OFFERING

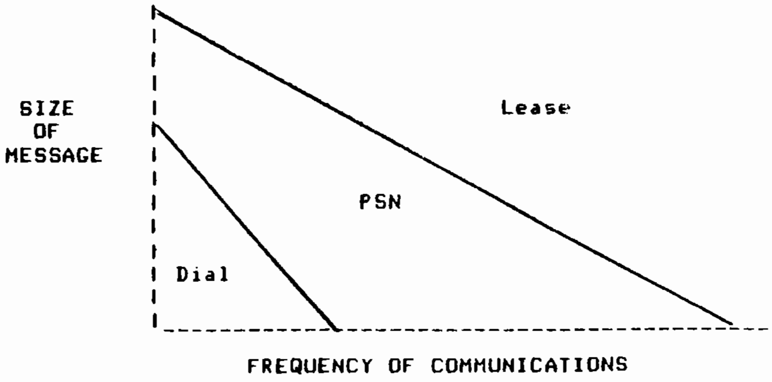
Just as X.25 networks have continued to proliferate and gain a greater importance in international data communications, so has HP's product offering. This paper will examine the emerging role of X.25 in HP's networking strategy by presenting an overview of X.25 technology, reviewing the current state of worldwide X.25 networks and services, reviewing the current HP product offering, and outlining the near term strategy for X.25 enhancements.

OVERVIEW OF X.25 TECHNOLOGY

Packet Switched Networks (PSNs) are a subset of the larger group known as "Public Data Networks". What they have in common is a particular technique for transmitting data from one station to another. Customer interest in packet-switching networks is due to the significant user benefits promised by widespread availability of these networks. The three most important benefits are:

1. lower data communications costs
2. increased network reliability
3. and a well-defined data communications standard.
4. simultaneous connection to multiple systems.

PSNs offer an alternative and potentially lower cost structure for data communications because they charge the user a flat monthly connection fee plus incremental charges based primarily on the volume of data the user transmits through the network. Charges are not strongly dependent on connect time or the distance between communicating systems, as is true for non-packet networks today. (On most PSNs, the charges depend exclusively on data volume; a few networks have a nominal connect and/or zone charge.) The telephone system, for instance, charges customers a flat rate for a lease telephone line, or a (time) X (distance) rate for dial-up circuit regardless of the amount of information the user actually transmits across the line. This means that if you use the telephone links to connect two computers you pay the same rate whether or not you actually send any data. With packet switching you pay primarily for what you send.

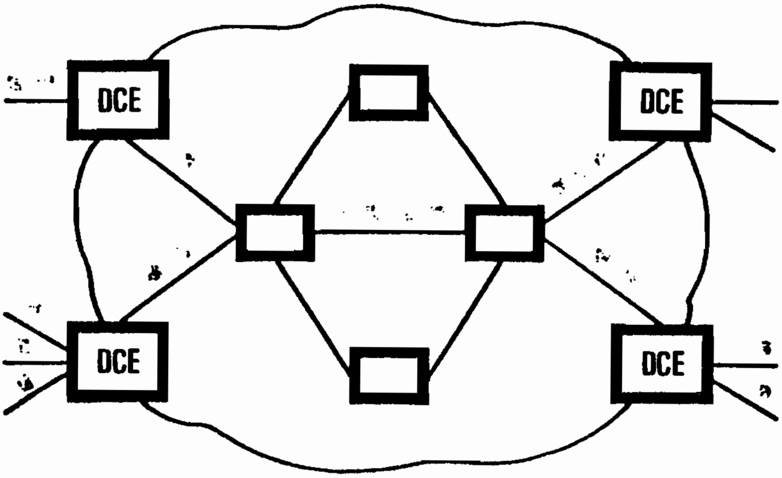


To understand how packet-switching networks can charge less money to provide the same data transport services as existing public networks (e.g.) the telephone companies or PTTs), requires a bit of insight into how the PSNs operate.

The first important thing to understand is that all information passing over a PSN must be broken up into fixed-length "packets" (hence the name of the networks). Since this packetizing function is not a user-level the length of these packets is unimportant to this discussion and may in fact vary from network to network. Even though the user may wish to send large contiguous blocks of data, it is important to realize that in order to be efficiently transported across the network, these large blocks must be broken down into smaller chunks - packets - by the data communications software in the system. This packetizing requirement is due to the fundamental difference between PSNs and other types of networks.

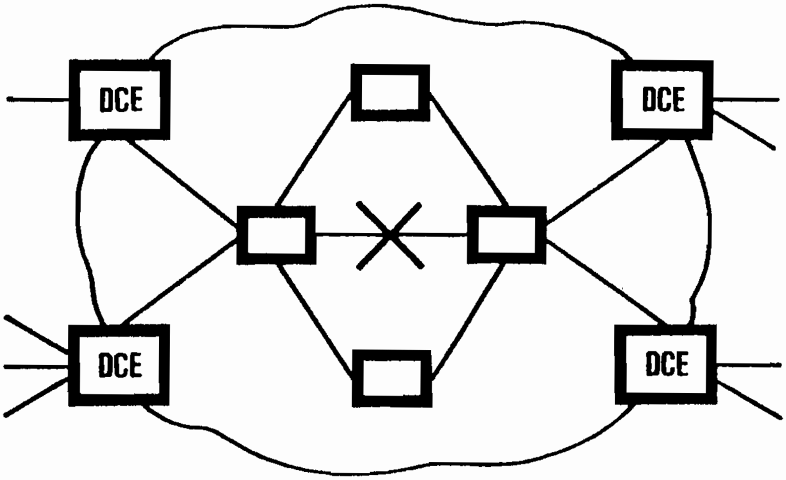
In today's public networks, a user 'monopolizes" a physical connection between source and destination. This is a cause for toll charges being keyed to circuit usage rather than data quantity. For example, in a leased circuit, only the users at each end of the link have access to the circuit. Similarily, in a dial-up connection, once the connection is made, only the two connected parties can use the physical wires connecting them. In these cases the parties involved must pay for the entire cost of the their physical circuit for the duration of their exclusive use. In PSNs, no one user has restricted access to any physical assets of the network. By not assigning physical assets to users, the network can multiplex many users' packets over the same wires and since the active path is composed of multiple short segments a single packet never occupies the entire length of the circuit from source to destination. The network can even go as far as to send packets aver multiple physical circuits as it attempts to maximize the use of its transmission facilities. This shared use of facilities is the basis for the potential cost reductions for any individual user.

COST: SHARING LINE BANDWIDTH



The second benefit of packet switching networks, increased network reliability, is a result of the the built-in redundancy provided by the basic network structure.

RELIABILITY: ALTERNATE PATHS

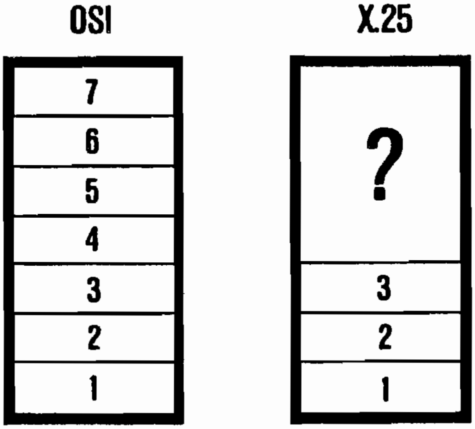


Note that this redundancy is done internal to the packet-switching network and is unimportant to the user, just as telephone company switching algorithms are unimportant to users as long as the calls are successfully completed. Different networks will undoubtedly perform this function using a variety of alternatives.

Typical PSNs are made up of Data Circuit-terminating Equipment (DCE) nodes which are the ports into the network and provide the user interface, plus internal switching nodes that route packets around the network. The highspeed connecting links internal to the network can be of any type including telephone system facilities, microwave channels, and satellite links.

As the figure illustrates, multiple physical paths exist between any particular pair of nodes in the network. The networks are able to recover from the failure of an internal node or communication link without disrupting end-to-end communication between users. They transparently re-route packets onto a link or through a node which is still functioning. Some very sophisticated networks even distribute internal data flows from heavily utilized links to underutilized links to permit maximum use of the network facilities. This capability is called "load leveling". In any event, users benefit from this network redundancy because it provides a higher availability level for their communications service.

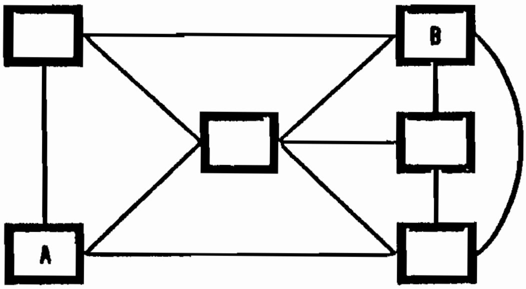
The third major benefit from packet-switching networks is that it is a standard (recommendation) developed by common carriers who provide transmission services. This standard is something that all computer vendors will have to support in their equipment in order to use the transmission services of the networks. In any environment where vendors all conform to a common standard, users will find it EASIER (but not automatic) to connect systems from multiple vendors together. To understand how this will make things easier for users we have to see how X.25 fits in with the broader standard under discussion.



As shown above, X.25 is actually only a portion of a complete layered communications arthitecture. The Open Systems Interconnection (OSI) standard is an international attempt to define an architectural model for data communications. The objective is to define a model to make data communications simpler and an effective and reliable tool for multi-vendor business data processing. Both standards are concerned with communications between machines and do not limit the implementation design to any system, so long as the interface provided conforms to the specifications. The X.25 model however only specifies the interface between a user's DTE (Data Terminal Equipment) which can be a terminal, computer, workstation, or other electronic device, and an X.25 network DCE, which is simply the physical port into the X.25 network. The OSI model attempts to define the layers, even within a system, all the way up to the user. The result is that X.25 has provided only the physical means for devices to connect and and communicate. (An analogy is with the telephone industry. With voice networks, equipment compatiblity has allowed a user in Edinburgh to connect to a user in Tokyo - but if they do not use (or at least understand) a common language, any data they exchange will be meaningless.) While, OSI attempts to define not only the connection means, but also the intelligence to interprete information.

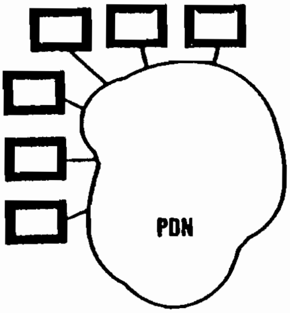
The fourth major benefit of X.25 networks is that from a single machine with only one physical connection into the network, one can simultaneously communicate with multiple systems. The ability to communication with multiple machines over a single line greatly increases configuration flexibility for large networks. There is no longer a requirement for a separate physical link to each system or a link through an intermediate node. PSNs reduce the number of connections needed to only one per system. Through that one link a system can access any other system on the network. Even more impressive is the fact that it is theoretically possible to have up to 256 simultaneous connections via one physical line. (The 256 is the number of outstanding packets that the software normally allows. User should note that the real number of simultaneous sessions allowed is dependent on the capacity that the line will support. For instance a 9600 bps line will support a maximum of 1200 characters per second. For 256 simultaneous sessions this implies an average of 5 characters per second per session, not a very efficient use of the line or the network.)

CONNECTING USER SYSTEMS
WITHOUT X.25 PDN



11 LINES + 22 CONTROLLERS

CONNECTING USER SYSTEMS
WITH X.25 PDN

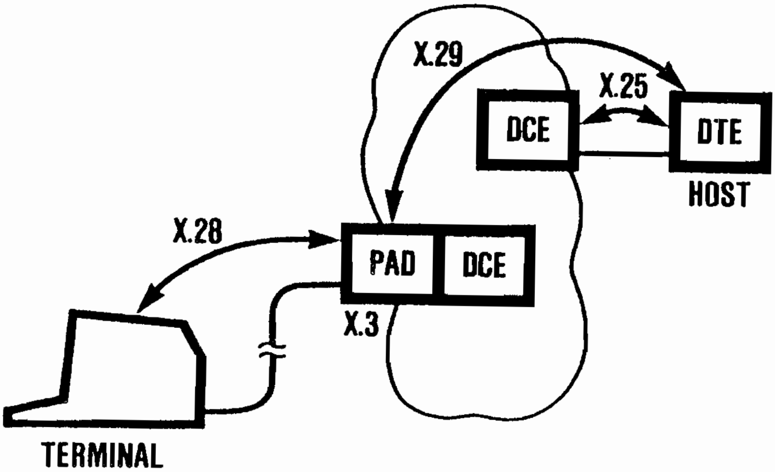


6 LINES + 6 CONTROLLERS

Since X.25 uses a full-duplex protocol, one users can be sending to one system while another user, on that one system is receiving information from a third system. This eliminates some of the response time problems associated BSC type protocols and alows for a much more efficient use of the communication lines.

To network administrators or planners this feature of X.25 networks provides savings from the use of fewer lines between nodes, fewer modems, and fewer system data communications controllers (INPs).

PSNs offer these benefits for both system-to-system and terminal-to-system communication. However, since most terminals are relatively non-intelligent devices, capable of asynchronous communications only, they they cannot communicate over PSNs in the same way that systems do. This has lead to the creation of additional standards for the support of terminals.



As shown above, PSNs have provided a facility called a PAD that acts as a front-end processor to the network port of entry (DCE). This allows users to connect asynchronous character mode terminals to the network, and then to a system. To handle terminal communications the networks have defined three additional standards. X.29 defines the protocol used between the host computer and the PAD; X.3 defines the capabilities that the PAD offers; and X.28 defines the protocol by which the user terminal converses with the PAD. When connected to the PAD terminals are generally limited to 300 or 1200 bps operation and character mode application support only. For many users this can be a major inconvenience from both a response time point of view and because of the lack of access to the more sophisticated block mode applications.

This brief discussion should give an understanding of the concepts associated with PSNs. Now lets move on to a discussion of the netwoorks that are available and some to the services offered.

X.25 NETWORKS AND SERVICES

Packet-switching networks have been operational since 1969 with the introduction of the ARPAnet designed for the US Department of Defense. This network was not available for general commercial use, but was used solely by research centers belonging to the Advanced Research Projects Agency. Since that time both public and private networks have continued to proliferate to a point where X.25 is internationally used as a means of transferring data. A list of countries with operational or planned networks is shown below. Although some of these countries can only boast of experimental links, several have matured to a point of widespread acceptance.

Operational Packet-switching Networks

- Argentina
- Australia
- Austria
- Bahrain
- Barbados
- Belgium
- Bermuda
- Canada
- Chile
- Dominican Republic
- France
- Germany (West)
- Hong Kong
- Israel
- Japan
- Kuwait
- Luxemborg (1983)
- Mexico
- Netherlands
- New Zealand
- Philippines
- Qatar
- Saudia Arabia
- Singapore
- South Africa
- Soviet Union
- Spain
- Sweden
- Switzerland
- Taiwan
- United Arab Emirates
- United Kingdom
- United States

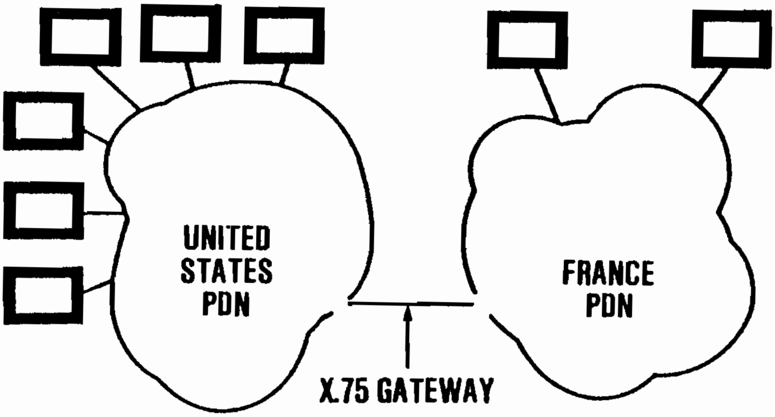
Planned Packet-Swiching Networks

- Brazil
- Denmark (1984)
- Greece (1983 via Euronet)
- Ireland (1984)
- Italy
- Korea
- Norway (1984)
- Portugal (1983 via Euronet)

As you can this represents an impressive list of countries providing packet-switching networks and the list is surely to grow. However, if the networks only provided the ability to transfer data within the geographic boundaries of the country the use of X.25 would be severely restricted. As we all know one of the major requirements in data processing is the distribution and dissemination of information among various groups and facilities within an organization. It is very likely that these groups and facilities are to locally not only remotely from each other, but also may be located in different countries. It is thus important that users of one PSN be able to place calls and transmit information to users on other PSNs. This is accomplished using gateways between the networks as shown below. In addition a patial list of interconnections is shown.

CONNECTING USER SYSTEMS

(WITH A PDN)



International Packet-switching Network Connections

	Austria	Belgium	Canada	France	Germany	Japan	Luxemb	Nether	Portuga	Spain	Sweden	Switz	UK	US
Austria		83		x	x					x		83		x
Belgium	x		x	x	83	x		83		x			x	x
Canada		x		x	x	x		x			x	x	x	x
France	x	x	x		x	x		83	x	x	x	x	x	x
Germany	x	x	x	x		x		83		x	x	x	x	x
Japan		x	x	x	x			x		x		x	x	x
Luxemborg	x	83		x									x	
Nether.	x	83		x	83					x			x	x
Portugal		83		x									x	
Spain	x	x		x	x	x							x	x
Sweden	x	x		x	x					x			x	x
Switz.		83		x	83	x				x			x	x
UK	x	x	x	x	x	x		83		x	x	x		x
US	x	x	x	x	x	x	x	x			x	x	x	

Rememeber that this is only a partial list of the connections available. Most PSNs have extensive connections to other networks with additional connections planned.

So far we have only discussed the ability to conect devices to a network; however, what one can do or has access to when on the network is of far more interest to users. In examining this question it appears that network resources are a varied as the networks themselves. All types of applications are available (either network supplied or vendor supplied). In particular user have access to:

- Conversational applications (interrogation or file updating, time sharing, transaction management, interactive file access, etc.)
- Data access with deferred transmission
- Remote Batch processing
- Interconnecting computers for file transfer or resource sharing.
- Transmission of messages, facsimile, electronic mail, etc.

In addition, some networks provide videotex (also known as viewdata) support. This service links special videotex terminals (or television sets) to computers so that information, stored and updated centrally on computers can be brought simply and cheaply directly into business and homes. Examples of networks offering the service are PSS in the UK, TRANSPAC in France, TELEPAC in Switzerland (1983), IBERPAC in Spain (1984), DATANET 1 in the Netherlands, LUXPAC in Luxemborg (1984), and Italy.

Other useful services provided by some networks are:

- Support for IBM 3270 terminal controllers (DATAPAC, TYMNET, TELENET, DATEX-P)
- Electronic banking funds transfers (Spain 1984)
- Terminal to terminal message switching (Spain)
- Electronic directories (France)
- Teletex support (Spain, Netherlands (1984), Luxemborg (1984), Ireland (1984), UK, Italy, etc.). Teletex is a CCITT regulated service that provides for communication between terminals used in the preparation and display of correspondance, and enables customers to exchange correspondence on an automatic memory-to-memory basis. An additional feature is that it should have interworking capability with existing telex networks.

Now that completes the section on network status; let's now take a look at HP's product offering.

HP'S X.25 PRODUCT OFFERING

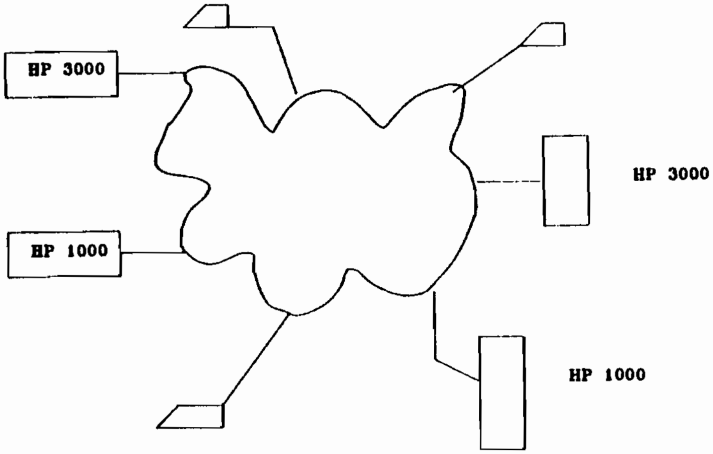
In October 1981, HP introduced the support of X.25 on the HP 3000 systems. This support was completely integrated into the DSN/DS software so that the user did not have to be aware of the type of connection in use when running DS communications. All DSN/DS services - Network File Transfer, Network File Access, Network Inter-Program Communications (PTOP), Network Database Access, Network Terminal Access, and Network Peripheral Access - could be used without any restrictions imposed be the type of connection.

In addition, support for the X.29 PAD standard was also introduced. This allowed the operation of asynchronous, character-mode terminals connected to the HP 3000. These terminals could either be connected via a lease or dial-up line and operated at speeds of either 110, 300, or 1200 bps.

This introduction permitted both system-to-system and terminal-to-system conversations to occur simultaneously over the same X.25 link.

Because X.25 support was integrated into DSN/DS, no modification or recoding of existing DS applications was required. No new hardware was required, only the INP. Thus if users wished to convert from lease, dial-up, or hardwired BSC connections to X.25 connections, all that was required was modems and connections to the packet swiched network. All existing DSN/DS users on CSS or SSS support contracts were automatically upgraded with the new X.25 software. In addition, all new DSN/DS users received the X.25 support with the installation of the DSN/DS software.

In February 1982, X.25 was announced on the HP 1000 computer family. This also provided for the full use of the DSN/DS facilities and for inter system communication between HP 3000's, HP 1000's, and asynchronous terminals via an X.25 network.

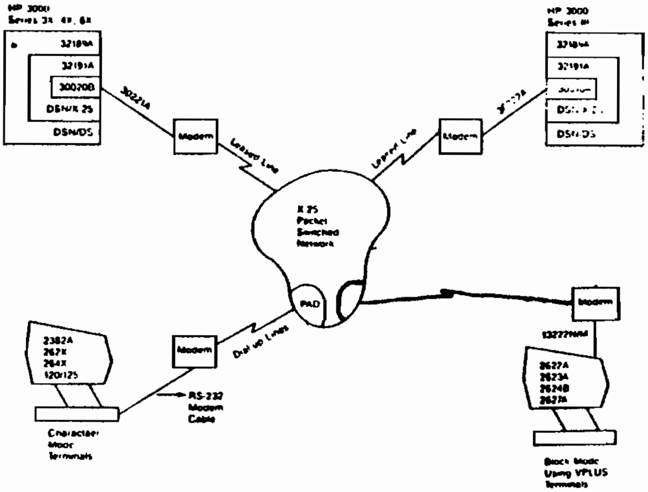


- SYSTEM TO SYSTEM COMMUNICATION VIA DSN/DS
- CHARACTER MODE TERMINAL SUPPORT
- HP 1000 TO NON-HP SYSTEM COMMUNICATION VIA X.25 STANDARD

Since that time, HP has significantly enhanced its X.25 offering with the following introductions.

VPLUS PAD SUPPORT

The largest requirement for X.25 connections is to connect remote terminals to a system or to multiple systems. This capability is provided by the networks (PAD support) but generally restricts users to character mode applications. To provide users better software compatibility, HP has enhanced VPLUS and created a new DSN/X.25 (32191A) product to allow support of VPLUS applications via X.25 networks.


HP 3000
Series 3X, 4X, 6X
32189A
32191A
30020B
DSN/X.25
DSN/DS
30221A
Modem
Leased Line
HP 3000
Series
32189A
32191A
DSN/X.25
DSN/DS
Modem
Leased Line
X.25
Packet
Switched
Network
PAD
Modem
13272NAM
2627A
2623A
2624B
2627A
Block Mode
Using VPLUS
Terminals
2382A
262X
264X
120/125
Modem
Dial up Lines
RS-232
Modem
Cable
Character
Mode
Terminals

This new product allows geographically dispersed terminals to access and communicate with multiple HP 3000 systems connected to the PSNs via the DSN/INP. It uses the X.3, X.28, and X.29 standards and supports transfer rates of up to 1200 bps. All HP character mode applications are supported on all HP terminals. In addition, HP 2622*, 2623*, 2624B*, and 2627 terminals support VPLUS (version B.03.15 or later) applications.

* These terminals require ROM updates to support VPLUS applications. consult your local HP Systems Engineer to insure that your terminals have be updated.

HP 2334A Cluster Controller

In September 1983, the HP 2334A X.25 Cluster Controller was introduced. The 2334A simplifies remote terminal connection to computers via X.25 networks by allowing up to 16 RS-232-C devices to be connected to the network via one comunication line. Connection to the network is done via standard modem leased line type connection furnished by the network. No special equipment or facilities are required. Terminals are connected to the 2334A through the use of a four port RS-232-C interface card. This card supports point-to-point devices at data rates up to 9600 bps with no modem control. Up to four of these cards can be installed in the 2334A giving excellent modularity to the product.

The intoduction of the 2334A provides users with increased data transfer rates to the network and lower connections costs for remote locations with the need for multiple terminals and printers, but unable to justify the installation of a computer.

If you remember the earlier discussion of PAD connections, we said that for asynchronous terminal connections user were required to obtain for the networks a special service for their terminals. Each terminal required one physical connection to the network and connection was at either 110, 300, or 1200 bps. In addition, most networks and computer X.25 software were not able to place outgoing calls to the PAD and thus printer connections were not possible. With the 2334A, sites with multiple terminals and printers require only one communication line with data transfer rates to the network being at speeds up to 19,200 bps. Higher speeds and lower costs are achieved for several terminals by sharing the cost of one communication line. A comparison of the two means of connecting terminals to the network is shown below.

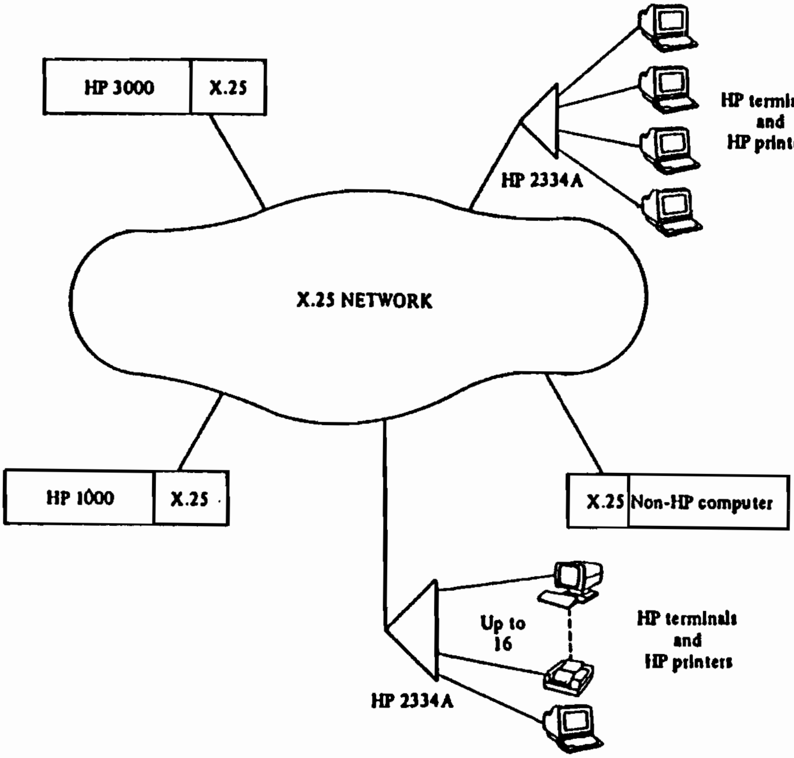
PUBLIC PAD	HP 2334A LINK
low speed connection	high speed connection
modem connection with limited error checking	highly reliable X.25 connection
no printer support	printer support

All HP terminals may be used with the 2334A for character mode applications; and obvioosly the 2334A is compatible with and supports the VPLUS enhancements that allows updated 2622A, 2623A, and 2624B and all 2627A terminals to execute VPLUS applications via the X.25 networks. In addition, the HP 250 computer system can be connected to the 2334A an act as a terminal in character mode or transfer files to the HP 3000 by using the TIO-II operating system of the HP 250. Finally, non-HP terminals using the X-ON/X-OFF protocol can be connected to the 2334A. However, it is the users responsibility to verify the correct operation of such devices.

With the 2334A, terminals can access to multiple computers simultaneously. In addition to computer-to-terminal connection via the 2334A, the same X.25 interface on the computer can be used to establish computer-to-computer connections and computer-to-terminal connections via the PAD.

Listed below are the features and benefits that the 2334A provides.

FEATURE	BENEFIT
* Remote connection of terminals over X.25 PSN.	* Low cost communication * Multiple CPU access via only one line. * Network reliability
* Certification with PSNs	* Use on national PSNs with possibility of international connection.
* Supports most PSN facilities	* Easier user access to PSN
* VPLUS support	* Transparent use of HP and customer application programs.
* Modularity of up to 16 ports in groups of 4.	* On site upgrade enabling addition of more terminals without renting of additional lines.
* High terminal throughput (16 x 9600 bps) and large buffer (32K bytes).	* Excellent response time with no character loss when all devices transmitting simultaneously.
* Statistical information on traffic available for host computer processing.	* Easier network management and planning.
* Symbolic name addressing of host computer.	* Simple user access to computer
* Full compatibility with X.3, X.28, X.29 standards.	* Possible connection to any computer supporting X.25/29.
* Flow control flexibility (ENQ/ACK and X-ON/X-OFF).	* Possible connection of wide range of devices.
* Default configuration in ROM.	* Automatic recovery from power failure. Unattended operation.
* Supports all 1980 CCITT X.3 parameters and additional HP proprietary PAD parameters.	* Powerful computer-to-terminal interaction.
* Customer defined private user groups.	* Reduced costs by suppressing need to use "closed user group" facility of the PSN.


HP 3000
X.25
HP terminals and HP printers
HP 2334A
X.25 NETWORK
HP 1000
X.25
X.25
Non-HP computer
Up to 16
HP terminals and HP printers
HP 2334A

The following table summarizes the connection possibilites and the system requirements for support of the 2334A.

COMPUTER	OPERATING SYSTEM	HARDWARE REQUIRED	SOFTWARE REQUIRED	APPLICATION SUPPORT
HP 3000	MPE-V MPE-VK	INP + cables	DSN/X.25	Character mode + VPLUS applications.
HP 1000	RTE-A RTE-6/VM	PSI + cables	DSN/X.25	Character mode applications only.
HP 250			TIO-II	Character mode + file transfer to HP 3000.
HP 100	Connected to HP 2334A as character mode terminal.			
Non-HP	Vendor supplied X.25/29 hardware and software required.			Character mode only. User responsibility to verify correct operation.

Network Certification

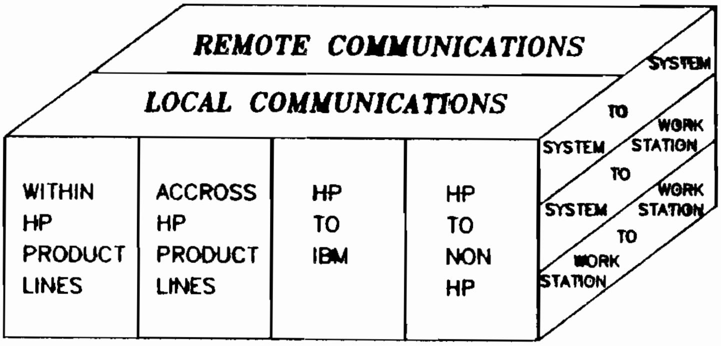
As you probably know, most networks require product certification before they can be used on the networks. This product certification can be anything from the submittal of a request form stating that your product meets the X.25 standards to extensive test to verify that the product conforms to the protocols and handles the network error routines properly. To satisfy these network certification requirements HP has develop a test program that exercises the system to verify proper operation on each of the X.25 networks. This exercise program has been distributed to trained field SEs and testing has either started or been completed on several networks. As additional SEs are trained the certification program will be expanded to more networks and countries. Shown below is the current status of network certification.

COUNTRY	NETWORK	HP 3000	HP 1000	HP 2334A
US	Telenet	Certified	Certified	Certified
	Tymnet	Certified	Certified	Certified
UK	PSS	Certified	Certified	Certified
Germany	Datex-P	Certified	Certified	Certified
France	Transpac	Certified	Certified	Certified
Canada	Datapac	Certified	Certified	*Not compatible
Netherlands	DN 1	Certified	Certified	Certified
Japan	DDX	Certified	Certified	
	Venus-P	Certified	In process	
Belgium	DCS	Certified	Certified	Certified
Spain	Iberpac	In process	In process	In process
Singapore	Telepac	In process		
Australia	Austpac	In process		
Argentina	Anpac	In process		

* Datapac is not compatible with the 1980 CCITT standard. Expect conversion to 1980 standard in mid-1984. At that time we will resubmit the 2334A for certification.

HP'S X.25 STRATEGY

This section will briefly outline the near term strategy for even better support of HP systems on X.25 networks. Keep in mind that due to the confidentiality of product development this discussion will be very general in nature. Also since products are under-development or in the investigation stage no dates on availability can be given.


REMOTE COMMUNICATIONS
LOCAL COMMUNICATIONS
WITHIN HP PRODUCT LINES
ACCROSS HP PRODUCT LINES
HP TO IBM
HP TO NON HP
SYSTEM TO SYSTEM
SYSTEM TO WORK STATION
WORK STATION TO WORK STATION

Dimensions of HP's Networking Strategy

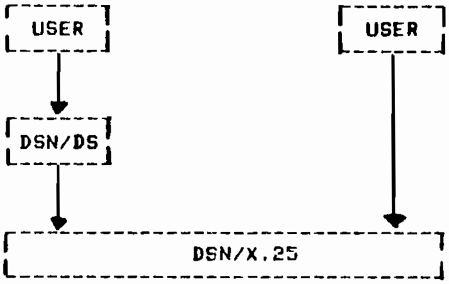
If you look at HP's general networking strategy you see that X.25 networks address the areas of remote connection of HP systems to other HP systems and HP systems to workstations. In addition, X.25 can be used as a standard for interfacing HP systems to non-HP systems. Now, how is this possible?

Remember X.25 is part of a broader standard intended to simplify the inter-connection of multi-vendor systems. Since X.25 provides a common protocol for communications and no more, users are required to develop additional upper level software in order for unlike systems to understand what is being passed across the communication line. These programs may need to address the data buffering, file format conversions, data format conversions, end-to-end reliability, etc. The task can range from simply developing cooperating programs to developing applications of the sophistication of DSN/DS. But first one needs access to the upper level of the X.25 standard.

If you look at the HP 1000 DSN/X.25 software that is exactly what has been provided. Users have the option of using the DSN/DS services to communicate between HP systems or developing applications that allow the HP 1000 to communicated to another vendors system (DEC, IBM, etc.)

The intention is to provide a similar capability on HP 3000s and longer term HP 9000 systems. This will make it easier (not automatic) to connect all HP processors to other vendor's equipment. REMEMBER FOR MEANINGFUL COMMUNICATION, USER LEVEL SERVICES MUST BE WRITTEN FOR BOTH MACHINES, such as program-to-program, file transfer, file access, virtual terminal, etc.

This enhancement will allow either the connection to other HP systems using the DSN/DS network services as supplied by HP or customized interfaces for use with other systems.



In addition to using X.25 as a means of interfacing non-HP systems, the other areas that we feel need to be addressed by our X.25 strategy are:

1. Support for applications other than VPLUS (HPWORD, TDP, HPSLATE, etc.)
2. Better support for HP personal computers.
3. Support for videotex terminals via X.25 networks.

This is by no means all, but we hope it gives you a feel for what we doing with X.25 and a realisation that X.25 is a major part of our long term communications strategy.

DATA COMMUNICATIONS FOR THE HP3000 USER

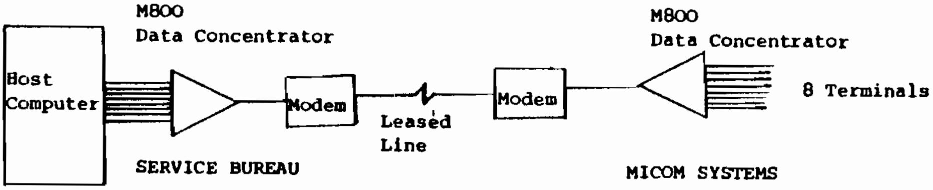
Micom is a rapidly expanding multinational manufacturer of data communications equipment for minicomputer users. Growth of the operation has been staggering. Three years ago Micom consisted of one operation situated in Chatsworth, Los Angeles employing 200 people. Now Micom have 5 buildings in the Chatsworth area and a number of flourishing subsidiaries in Puerto Rico (manufacturing) Pittsburgh PA (Black Box) Santa Clara CA (Microbaud) and Reading (Micom-Borer). Currently the totel headcount is in excess of 1000.

To meet the ever increasing demands of the organisations, considerable energy was invested in computerisation. Three years ago the company met its data processing requirement by using service bureau. Today there are six systems at Chatsworth, with other systems in Puerto Rico, Pittsburgh and Reading.

Soon after we began installing the systems in house we realised the need for local networking. We chose a data PABX approach and I believe time has shown that to be a wise decision. The system has easily grown to accommodate more machines and users. We think our experiences are typical of many growing organisations.

Evolution of Communications Requirements

Back in the early days Micom utilized the Xerox service bureau for accounting and manufacturing control. Terminals were linked into the bureau via telephone lines and modems. Leased lines were adopted due to the requirement of utilising the service for much of the working day. Dial-up access would not have been cost effective.



Remote Connection to Service Bureau

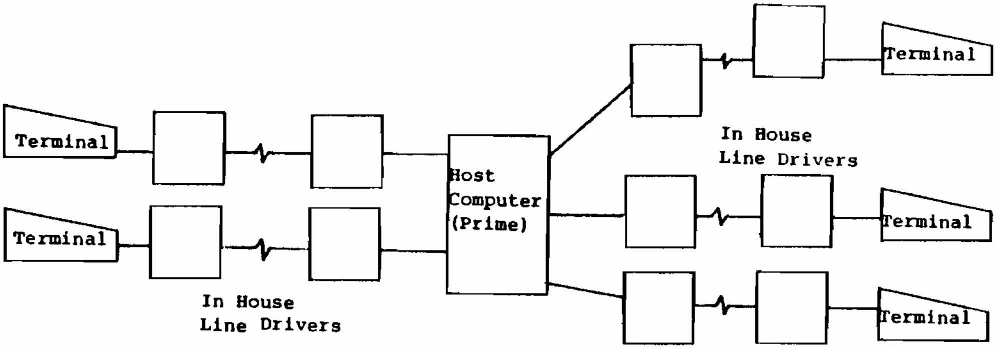
Figure 1

As the terminal population increased we installed 8 channel data concentrators at each end of the leased line. Such concentrators (statistical multiplexors) permitted upto 8 terminals to link into the bureau over a single leased line. Cost savings over a multileased line approach were considerable. Not only were line costs reduced but hardware costs and complexity were considerably less. Instead of 8 modems per end and 8 phone lines our link had only one modem and one concentrator per end. Using this approach cost savings were apparent in just a few months.

The First Computer System

Then, about 2½ years ago, our growing size and DP requirements finally sent us looking for our first in-house computer. We hoped for a single system that could handle all our DP needs: administrative, financial, engineering, software support, customer service and CAD/CAM. We did not choose an HP3000, as this audience might expect; we installed a Prime 750, mainly due to software considerations.

However, by the time the Prime was up and running, we had already outgrown it. We proceeded to order our first HP3000, a Model 44 (again chosen due to its software), along with another, smaller, Prime to handle CAD/CAM on a dedicated basis. By this time, we had decided to put financial and manufacturing applications, including MRP and Bills of Material, on the HP, leaving the Primes for use by our customer service and engineerng departments.



In House Communication Using Line Drivers

Figure 2

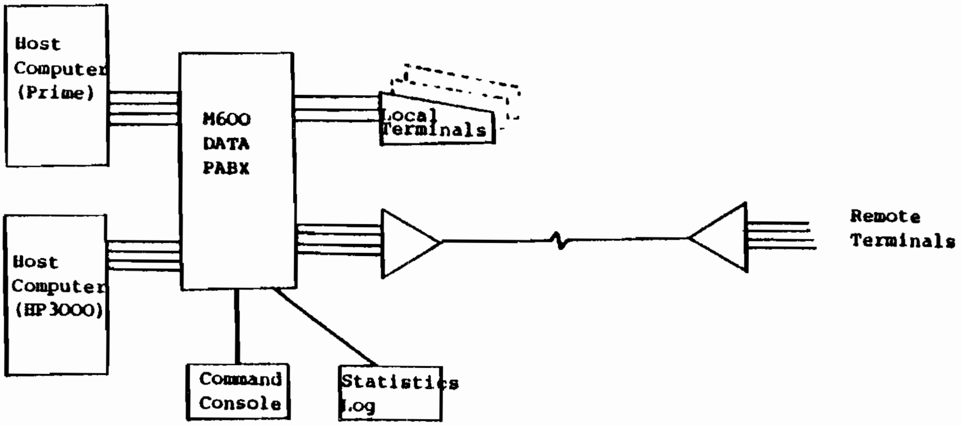
Fortunately for us, with the installation of our first computer, the Prime, we made a data communications policy decision which is with us today: unless the terminal is in the same room as the computer, we use line drivers to connect the terminals to their hosts. For those who are not familiar with them, line drivers are functionally analogous to modems but much less expensive. Capable of operating over distances of a few miles, they condition the signals going between computers and terminals in order to ensure reliable transmission beyond the EIA RS232C-specified maximum cable run of 50 feet.

Accessing Multiple Computers

When the second computer was installed, an HP3000, a new problem presented itself. While we had managed to split applications between HP and Prime systems, many of the terminal users needed to access to both.

Faced with providing one user access to two computers, we had several choices. We could have added extra wires through walls and ceilings for each multi-machine user, and asked the user to plug his terminal to the appropriate set as needed, or we could have -- even more magnanimously -- installed a second terminal for each of these folks. Neither solution was economically viable, particularly when we envisioned the consequences of adding more people and more machines to cover our continuing growth.

Instead, we left the wiring and terminal situation as it was, and made our move in he computer room, where we installed a data PABX. With the data switch, as it is often called, we have in essence a private telephone system that can connect any authorised user to any available computer port. And, if a port is not available, say all the HP ports are tied up with accounting business at the end of the quarter, then the user asking for an HP port is told (in effect) "They're all busy. Want to wait? You'll be number three in line." This is the data PABX equivalent of camp-on busy in a voice exchange.



M600 Data PABX System

Figure 3

The Advantages of a Data PABX

The primary advantage of a data PABX, although it has many, is its ability to connect any terminal to any requested resource (subject to security considerations programmed into the switch). This means we can hardwire a terminal to the switch. This means we can hardwire a terminal to the data switch, and from the terminal keyboard a user can request a port on one of our HP systems, one of the Primes, a Zilog development system our engineers use, or whatever we may acquire in the future. We can add more computer systems and terminals as we grow. We can even let users call outside, through the switch, to reach one of the service bureau.

We also benefit from a 'statistics log' feature of the switch which provided detailed reports of all switch activity (a function which would require a dedicated processor in other proposed networking plans). And our network easily supports remote users and remote computers.

The switch also lets us save money by using leased lines, while still providing 'dial-up link access' to several computers. Nothing keeps us from 'remoting' some of the CPUs, heightening that dial-up parallel.

An example may help to illustrate the degree of the flexibility we realize by combining data concentrators, leased lines, and the switch. Say one user at Micom Caribe, our Puerto Rican facility, needs to use one of the HP systems. He simply turns his terminal on, and the switch, which is in our headquarters in Chatsworth, California, automatically asks him where he wants to be connected. He can answer with a symbolic name, in our case with an 'H' for one of the HPs or with an 'M' for the HP running MANMAN, or whatever, and the data PABX makes the connection.

A few minutes later, an engineer in Puerto Rico may turn on his terminal and ask for one of the Primes. Although both active terminals are on the same leased line, each user gets the same service he would on his own unshared line. For that matter, either of these two users can log off one machine and ask for another without affecting the transmission of the other user on the line.

Perhaps it's inevitable that our version of local networking using a data PABX is continually compared to others which use more exotic technologies. We don't mind. We show up well in the comparisons. For example, the new proposed local networks which use coaxial cable or fiber optics offer one very appealing feature in their ability to support many users on the same physical medium. But we can do the same by adding a few new twists to plain old telephone technology. In our corporate headquarters we have begun using a new type of line driver that can multiplex up to eight asynchronous terminals over the same two pairs of wire that might otherwise be used for a single telephone or terminal. This saves us time, money, and effort. We already have our offices wired for terminals, so now we can use the same wiring to connect several terminals in one room to the data PABX. In addition to saving us the time and cost, and disruption of stringing new wires, it also reduces the number of line drivers we need by as much as a factor of eight.

Even now the necessity of stringing additional cables around the building for data purposes is becoming dated. Recently we've been using an approach where the installed telephone system wiring can be utilised for both voice and data purposes. Here the data is summed with the voice input into a box which resides under the telephone. The data channel is then extracted at the voice PABX interface and 'piped' to the data PABX. This approach virtually eliminates the wiring overhead normally associated with local networking and allows a terminal to be connected wherever an extension telephone is sited.

Expanding the System

As might be expected, as our data processing capabilities have grown, so has the data PABX. Luckily, the switch is easy to expand with the addition of simple plug-in interface card modules. These provide four line or port interfaces per card slot, and each bay has up to 32 slots, which works out to 128 lines or ports (intermixed) per 19-inch bay.

As we fill a bay, we simply add another. Once we had our first HP3000 and Prime pair connected to the switch, it took about 30 days before we added our second bay; 60 days later we put in bay number three. We've already ordered our fourth, and we can continue to put them in until the raised floor collapses from the weight.

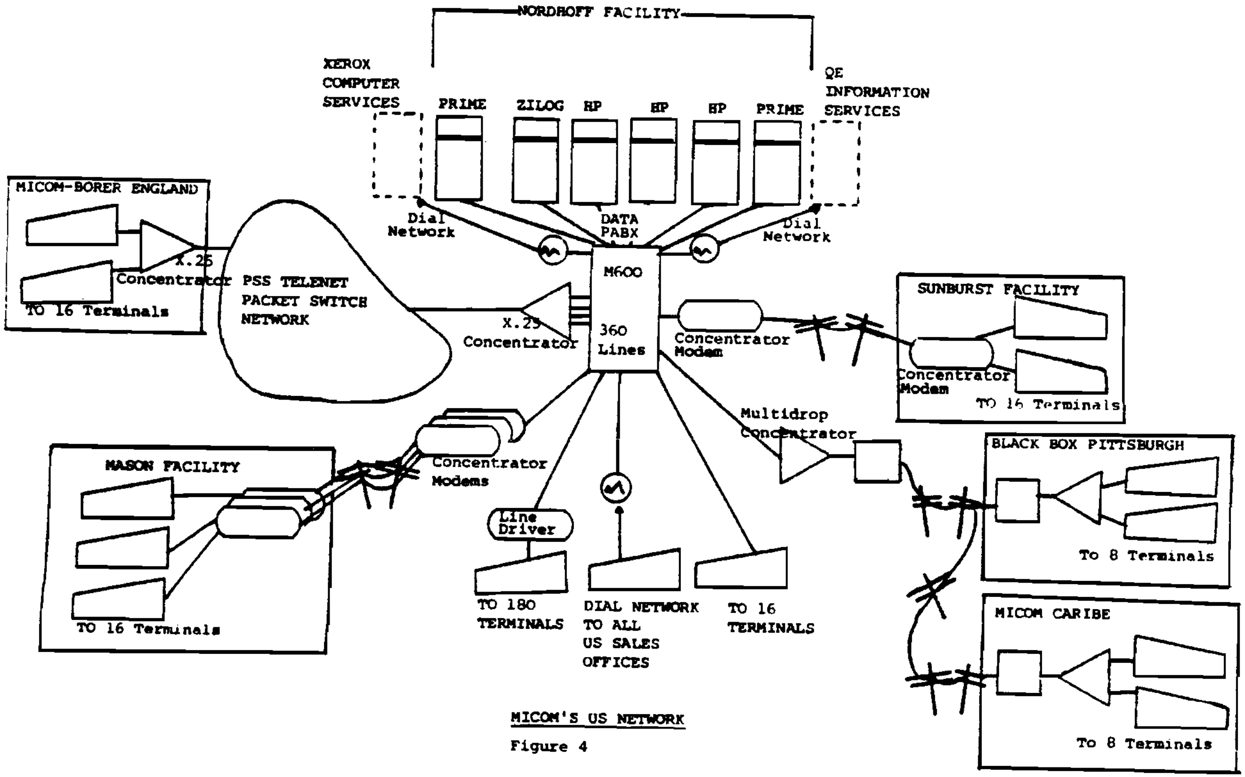
All of this can lead to a massive set of cables for local and remote terminals, we found. We have about 250 terminals in our Chatsworth facilities, and another 30 to 40 in the field. To simplify the wiring of the nearly 300 RS232C connections coming into the computer room, we have adopted a technique called 'group termination.' All incoming lines go to a wall-mounted telephone block; cables with 50-pin connectors attach the block to the switch. Each of the cables handles either six terminals (with EIA control signals) or 12 terminals (data only).

Of course this greatly reduces the snarled wiring behind our switch, and decreases the cabling under our flooring. It also makes connecting to the data PABX guicker, as we can attach as many as 12 terminals when we plug in a single 50-pin connector. (The group termination technique we use should not be confused with another that uses similar wall-mounted blocks but omits the 50-pin connectors, leaving the user with a fistful of low wires that must be connected to screw terminals on the switch).

Expanding the Network

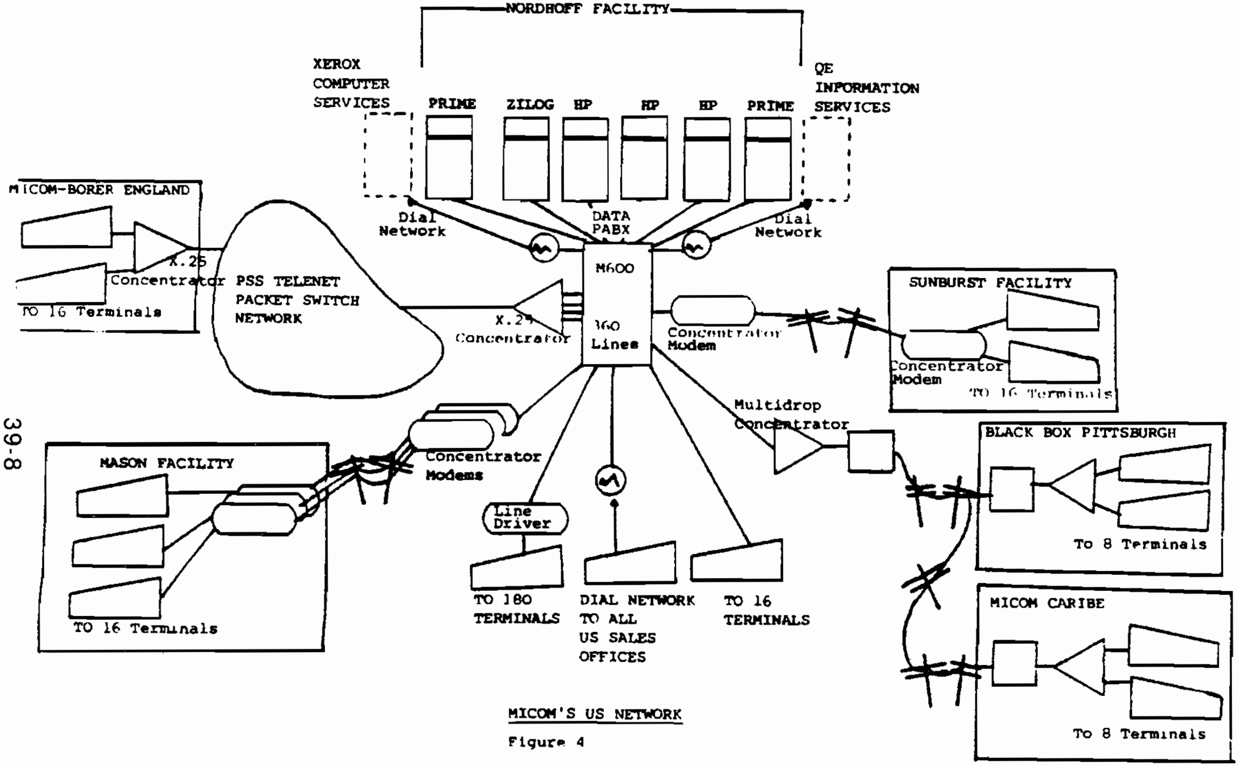
As mentioned earlier the Micom network is just restricted to operations in Chatsworth. Micom Caribe now run a HP3000 model 64 for accounting and manufacturing control purposes. Terminal access is accomplished in a similar way to that described in Chatsworth. Much more recently a HP3000 Series 64 has been installed in Black Box at Pittsburgh Pennsylvania. Both these sites are linked into Chatsworth via a shared leased line. This in its own rights is an unorthodox approach to asynchronous working but thanks to Micom's 900 Multidrop Concentrator line saving can be achieved where the geographical topology is appropriate.

In July this year Micom-Borer in Reading installed a HP3000 Model 40. Micom-Borer is a standalone sales and manufacturing operation and hence has a need for implementing all the standard accounting and manufacturing disciplines - basically a subset of Chatsworth. Linking into Chatsworth from the UK via a leased line is not a cost effective proposition at around £35000 per annum. Up until now we have utilised dial-up services with the adoption of a 1200bps V.22 modem. Typically error probability is high when dialling over 6000 miles so we have utilised M500 error controllers to permit error free operations. Now with the advent of the PSS X.25 service we are experimenting with connection via Telenet into Chatsworth. Even though the service is ostensibly X.25 we are interfacing the service with X.25 PADs that permit 8 asynchronous terminal channels to connect over a single X.25 connection. The results so far are very encouraging bearing in mind the relatively small amounts of data transfered.


NORDHOFF FACILITY
XEROX COMPUTER SERVICES
PRIME
ZILOG
HP
HP
HP
PRIME
QE INFORMATION SERVICES
Dial Network
DATA PABX
Dial Network
MICOM-BORER ENGLAND
X.25
Concentrator
TO 16 Terminals
PSS TELENET PACKET SWITCH NETWORK
X.25
Concentrator
M600
360 Lines
Concentrator Modem
SUNBURST FACILITY
Concentrator Modem
TO 16 Terminals
Multidrop Concentrator
BLACK BOX PITTSBURGH
To 8 Terminals
MICOM CARIBE
To 8 Terminals
MASON FACILITY
TO 16 Terminals
Concentrator Modems
Line Driver
TO 180 TERMINALS
DIAL NETWORK TO ALL US SALES OFFICES
TO 16 TERMINALS

MICOM'S US NETWORK

Figure 4


NORDHOFF FACILITY
XEROX COMPUTER SERVICES
PRIME
ZILOG
HP
HP
HP
PRIME
QE INFORMATION SERVICES
Dial Network
DATA PABX
Dial Network
MICOM-BORER ENGLAND
X.25
Concentrator
TO 16 Terminals
PSS TELENET PACKET SWITCH NETWORK
X.25
Concentrator
M600
360 Lines
Concentrator Modem
SUNBURST FACILITY
Concentrator Modem
TO 16 Terminals
Multidrop Concentrator
BLACK BOX PITTSBURGH
To 8 Terminals
MICOM CARIBE
To 8 Terminals
MASON FACILITY
TO 16 Terminals
Concentrator Modems
Line Driver
TO 180 TERMINALS
DIAL NETWORK TO ALL US SALES OFFICES
TO 16 TERMINALS

MICOM'S US NETWORK

Figure 4

The Next Step in Networking

We're also expanding the scope of our networking in another way. Our HP3000 in Chatsworth, will be expected to support DS/3000. The new machine will connect to our data PABX and through a high speed link to one of the other 3000s. Our reason for going to DS/3000 is one of response time for our many users in finance, manufacturing, and sales support who make inquiries against a large IMAGE data base now residing on one of the 3000s.

Initially we expect to put MANMAN and Accounts Payable on one of the linked 3000s, General Ledger, Accounts Receivable, and Order Management applications on the other. Files unique to each host will become more readily available to our users, and common files will be kept synchronised through the facilities of DS/3000.

Because of the transparency of the data PABX, we don't foresee any problems with the two networking systems operating together. Additionally, if someone needs to switch applications, say from A/R to A/P, it will be more efficient to reconnect to the appropriate host through the switch, as opposed to using the high speed interprocessor link and stealing capacity from DS/3000.

Network Conclusions

As we've grown our network, we've learned a few things, the most important of which is just how right we were to start with a data PABX. Unless your users stick to a single computer, or don't mind marching off to a centralised terminal room, it's difficult to see how to manage without one.

Granted, telephone technology is not as exotic or glamorous as working with coax or glass fibres, but it more than makes up for its lack of sexiness. Its technologies are proven, and relatively standardised in a de facto sense. Twisted-pair wiring or in-house telephone wiring is inexpensive, and in most offices, it's already in place. Likewise, using line drivers with RS232C interfaces provides a standard method of connecting to the network and avoids any special programming considerations.

In contrast, networks using coaxial cable or fibre optics run up increased costs due to the expense of the medium. In most buildings the installing of the broadcast medium also runs costs up very quickly, as well as disrupting everyone's work. Then too, the interfaces to the media are more expensive, by an order of magnitude, than line drivers, and are basically unstandardised, incompatible devices today.

A pioneering user who adopts one of these new and exotic networking technologies may well be casting his decision in concrete before the industry is ready for that. A mistake on his part is likely to become an expensive embarassment down the road. We're not going to have that problem. When those new technologies mature and users can install them with confidence, we won't be left behind; we can connect to them too.

Communications Hardware Glossary

The previous sections have illustrated Micom's approach to building a corporate network encompassing local networking, and distant networking techniques. Let's analyse the key products that allow such networks to be implemented.

Local Networking Products

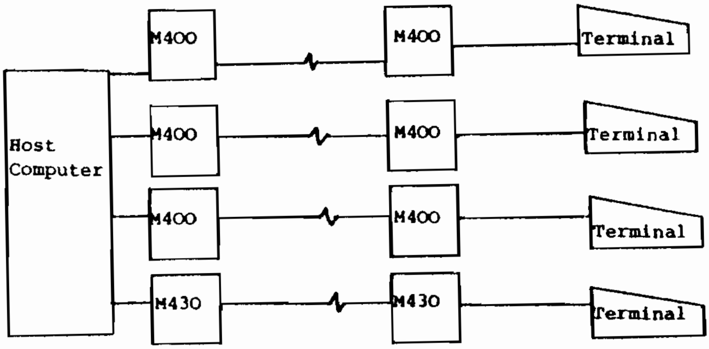
Micom's approach to networking is generally called 'Instanet'. What comprises 'Instanet'?.

MICRO600 PABX

The Micro600 Series 2A Port Selector provided contention, permitting a large number of asynchronous terminals to compete for access to a smaller number of computer ports, and port selection, allowing the terminal operator to select from his keyboard a particular 'class' of computer port. Upto 1504 terminals and/or computer ports can be attached to a single Micro600

M400 Line Driver

The Micro400 Series are a range of specialized modems designed specifically for short haul data transmission offering substantial cost savings because of their reduced complexity. They operate asynchronously or synchronously over customer - owned twisted pair cable and are suitable for a range of applications including point-point and multipoint operation, at speeds upto 19,200bps.

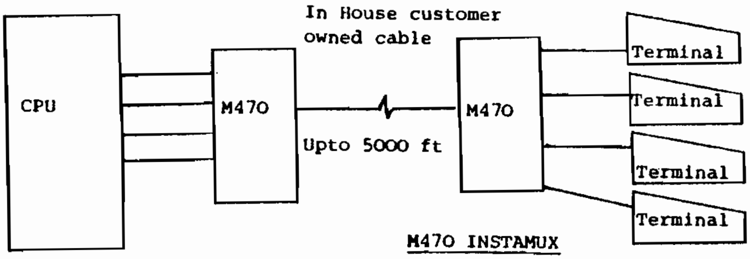


Micro400 Line Driver

Figure 5

M470 Instamux

The Instamux 470 is a low cost, high performance, asynchronous data multiplexor with built-in line driver which operates over low cost customer-owned cables. It significantly reduces customers installation costs by providing a path for a cluster of up to eight terminals over a single cable. In a new site the customer saves money by installing fewer cables. In an existing computer site the customer saves money by being able to expand the number of terminals without adding new cables. A cable that initially was used for only one terminal can now be used for up to eight terminals with no difference in response time for users.

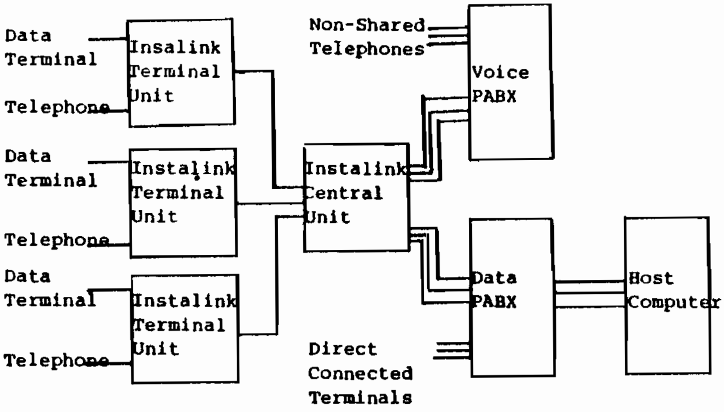

In House customer owned cable
CPU
M470
Upto 5000 ft
M470
Terminal
Terminal
Terminal
Terminal

M470 INSTAMUX

Figure 6

M460 Instalink

Micom's Instalink Voice/Data Multiplexor provides a convenient solution to instant local data networking. It is installed by simply being plugged in between the users personal terminal and his telephone handset, the device 'piggybacks' data transmission onto voice transmissions enabling the users internal 2-wire PABX telephone system to become a local data network. With Instalink, every desk with a phone can have a computer data terminal without expensive rewiring.


Data Terminal
Telephone
Insalink Terminal Unit
Data Terminal
Telephone
Instalink Terminal Unit
Data Terminal
Telephone
Instalink Terminal Unit
Instalink Central Unit
Non-Shared Telephones
Voice PABX
Data PABX
Direct Connected Terminals
Host Computer

M460 Instalink System

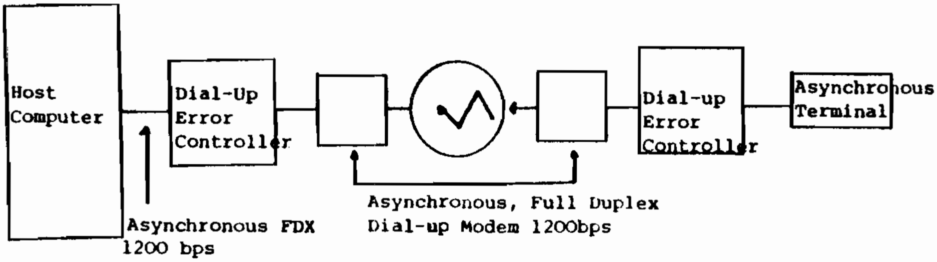
Figure 7

Distant Networking Products

The adoption of appropriate products for distant networking depends solely on the application to be fulfilled. A number of products however are particularly pertinent to the HP3000 user.

Micro500 Error Controller

A Micro500 unit is installed at each end of the data link, between the data terminal and the modem at one end and between the computer port and the modem at the other end. It is designed for use in a wide variety of asynchronous terminal applications, primarily offering the same retransmission-on-error capabilities as the Micro800/2 but for single terminal installations. It can also operate as a synchronous-to-asynchronous converter allowing asynchronous terminals to be used with synchronous modems. The Micro500 can be used as a full duplex to half-duplex protocol converter allowing asynchronous terminals to operate with half-duplex synchronous modems.

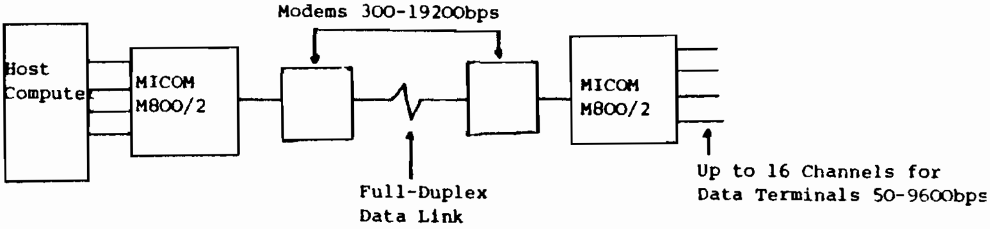

Host
Computer
Dial-Up
Error
Controller
Dial-up
Error
Controller
Asynchronous
Terminal
Asynchronous FDX
1200 bps
Asynchronous, Full Duplex
Dial-up Modem 1200bps

Micro500 Error Controller

Figure 8

Micro800/2 Data Concentrator

The Micro800/2 Data Concentrators are a range of low cost statistical multiplexors designed to allow up to 16 terminals, synchronous and asynchronous to share a single telephone acheiving immediate savings in telephone line costs even when supporting just one CRT and a printer. They require no changes to existing hardware and software and provide each terminal with an apparent direct connection to its host computer.


Modems 300-19200bps
Host Computer
MICOM M800/2
MICOM M800/2
Full-Duplex Data Link
Up to 16 Channels for Data Terminals 50-9600bps

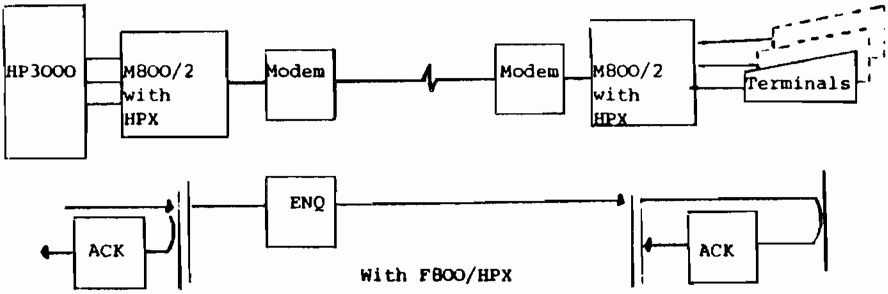
Micro800/2 Data Concentrator

Figure 9

They also provide extra advantages including automatic retransmission on error and easy do-it-yourself troubleshooting. Terminals may also use dial-up access to the remote concentrator if required. The data link between the Micro800/2s must be point-to-point and full duplex but may operate with asynchronous modems to 9600bps or synchronous modems to 19,200bps. The modem speed required depends on anticipated terminal activity but is typically one-half to one-quarter of the aggregate channel rate.

F800/HPX

The F800/HPX option for the Micro800/2 eliminates the problem of long delay times suffered when using data concentrators with HP3000 computer by emulating the ENQ/ACK protocol at the interface to both the host and terminal. With the F800/HPX option, Micom's Micro800/2 ACKs block mode messages at the host and obeys the ENQ/ACK protocol at the terminal.

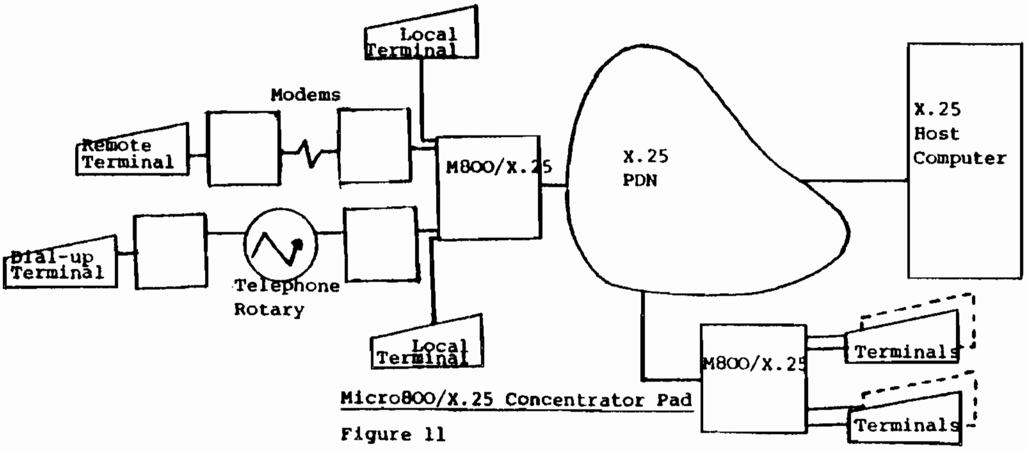

HP3000
M800/2 with HPX
Modem
Modem
M800/2 with HPX
Terminals
ACK
ENQ
ACK
With F800/HPX

Micom F800/HPX Option

Figure 10

Micro850/X.25 PAD

The Micro800/X.25 Concentrator PAD is an X.25 packet assembler/disassembler which concentrates the asynchronous data from four to twenty-four terminals or computer ports onto a single X.25 access line. It meets all the requirement of CCITT recommendations X.25, X.3, X.28 and X.29.

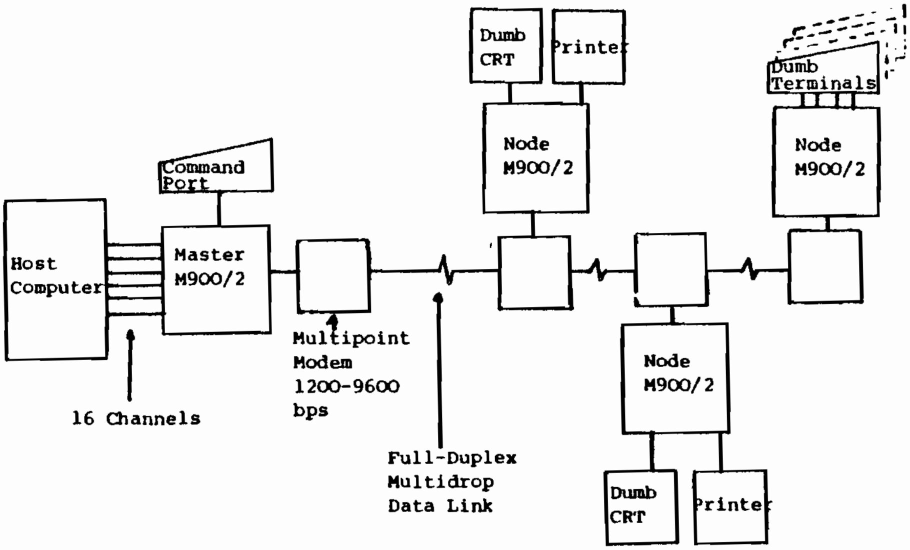

Local Terminal
Modems
Remote Terminal
M800/X.25
X.25 PDN
X.25 Host Computer
Dial-up Terminal
Telephone Rotary
Local Terminal
M800/X.25
Terminals
Terminals

Micro800/X.25 Concentrator Pad

Figure 11

Micro900 Multidrop Concentrator

The Micro900/2 Multidrop Concentrator's primary use is in supporting scattered remote terminal populations whose computer acces is over telephone lines. In situations where each site has a cluster of terminals, the Micro900/2 system may be substantially less expensive than point-to-point data concentrators systems due to its great advantage in telephone line savigs. It can be used for linking up to 16 regional offices with a central computer and provides automatic retransmission-on-error, fault isolation capabilities, greater efficiency in line use and higher speed transmissions, requiring no changes to computer hardware or software.


Dumb CRT
Printe
Dumb Terminals
Node M900/2
Command Port
Host Computer
Master M900/2
Node M900/2
Multipoint Modem 1200-9600 bps
16 Channels
Node M900/2
Full-Duplex Multidrop Data Link
Dumb CRT
Printe

Micro900/2 Multidrop Concentrator

Figure 12

K G Baynton

15th August 1983

SIMPLIFY

WORKSTATION TO HOST

CONNECTION

HEWLETT - PACKARD
GRENOBLE NETWORKS DIVISION

How can a large number of terminals be connected simply and efficiently ? How can you connect terminals scattered across wide areas or in different parts of a large building ? Can 'dumb' terminals be connected in the same way as personal computers ? What kind of support should a user expect for an installation involving many different vendors ?

Transmission of different types of informations and communications - voice, graphic image, computer data and so forth - is a field that has grown rapidly and will expand even faster in the future. The telephone system is a good example of how multiplexing technics have evolved in the past. In order to reduce the cost of communication, telephone companies have developed elaborate schemes for multiplexing many conversations on a single physical channel. Better usage of the bandwidth available on the media permits cost to be shared by more users, thus offering the same service at lower cost to each individual users. The same phenomenon is occuring in the area of data transmission.

The intention of this paper is three-fold. First, it will describe the historical evolution of multiplexing technics. Second, it would present and compare data communication protocols and analyze various statistical multiplexers or stat-muxes including their functional descriptions, features and benefits. Third, it will present the HP solution to multiplexing, through our family of cluster controllers, the HP233x.

Multiplexing technics

The first multiplexing scheme used was **Frequency Division Multiplexing (FDM)**. In FDM the frequency spectrum is divided up among the logical channels, with each user having exclusive use of his frequency band. AM radio broadcasting provides an illustration of FDM. Different frequencies are allocated to different logical channels (stations) each operating in a portion of the total spectrum (about 1 Mhz) allocated for AM broadcasting. The interchannel separation is sufficiently large to prevent interference between two adjacent channels.

Figure I shows how three voice-grade telephone channels are multiplexed using FDM. The same principle is used to multiplex data equipment onto a voice-grade channel, frequencies and bandwidth differ. In order to keep the channels well separated, a bandwith larger than the usable bandwith is allocated to each channel. First, channels are raised in frequency, each by a different amount. They can ten be combined, since no two channels occupy the same portion of the spectrum.

For voice FDM, a widespread standard is twelve 4000-Hz voice channels (3000 Hz for the user plus two guard bands of 500 Hz each) multiplexed into the band 60 to 108 KHz; as for Data FDM, the standards define twelve 150 baud channels (each channel using 240 Hz, including guard bands) multiplexed onto a 3000 Hz voice-grade bandwidth.

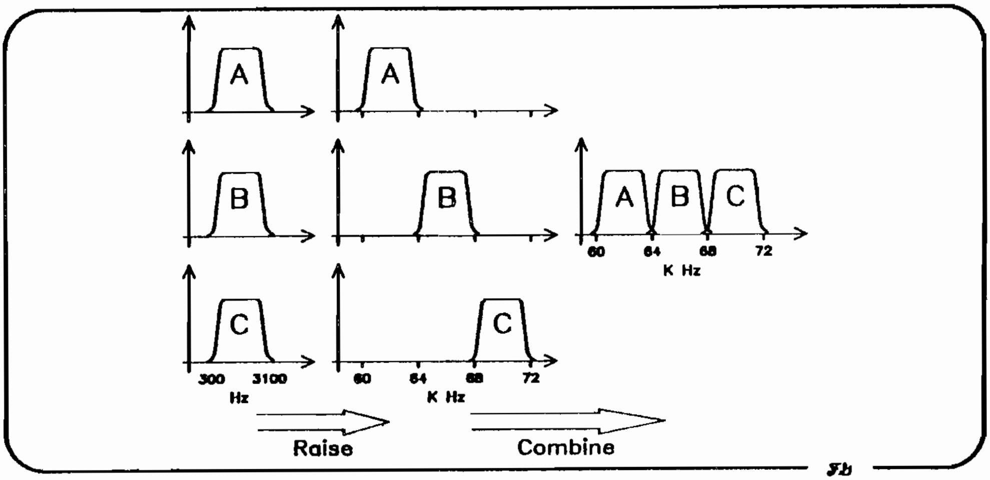


Fig. I: FDM of three voice-grade channels.

Notice that even though there are gaps (guard bands) between the channels, there is some overlap between adjacent channels, because the filters do not have sharp edges. This overlap means that a strong spike at the edge of one channel will be felt in the adjacent one as non thermal noise.

FDM has no ability to detect and recover transmission errors. It is also not flexible because it does not allow reconfiguration (adding or changing characteristics of a channel) easily. A good feature of FDM, however, is its ability to build multidrop networks. Since each channel uses a different frequency within the bandwidth, individual channels can be dropped and inserted at different points along the same telephone line.

From FDM examples above, it is quite obvious that a fair amount of the bandwidth is wasted by the guard bands. This method took advantage of technics and equipment used in radio, where frequency manipulation were common. However new multiplexing schemes can divide the transmission path much more precisely, thus being more efficient. Instead of dividing the spectrum by frequencies, each user is assigned the total bandwidth of the media during a time slot. This is **Time Division Multiplexing (TDM)**.

Time division multiplexing can be compared to a switch that rapidly samples a number of lines. These samples are sent across the data link, then routed back to their original sequence by another switch at the receiver. Figure II illustrates this method.

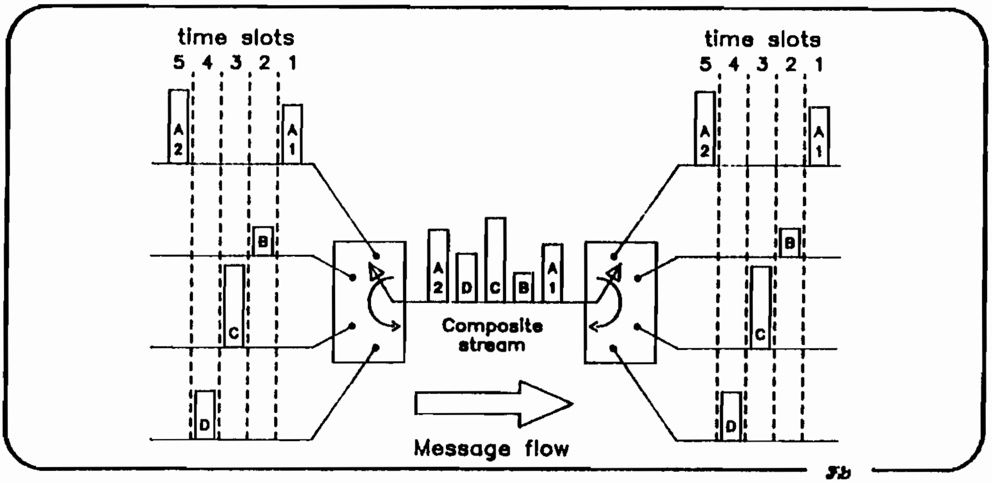


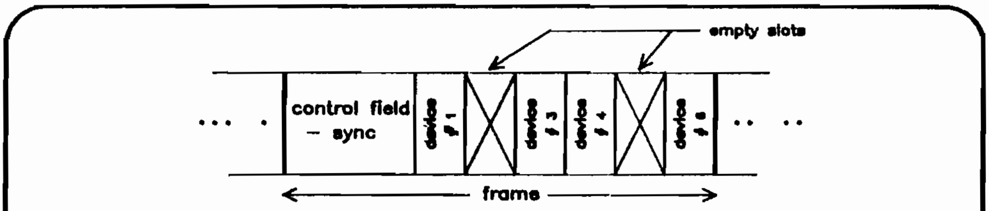
Fig. II: TDM principle.

Data from each channels is grouped into 'frames' preceded by a control sequence that permits synchronization between the two multiplexers. Frames are transmitted continuously and contiguously, the receiving end knowing which data belongs to each channel as a function of their time relationship to the begining of the frame. The data carries no identification of its origin. Figure III (a) shows an example of a TDM frame. Since the TDM transmits continuously, there is no possibility of retransmission even if the receiving end detects errors.

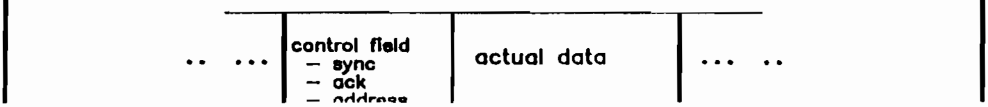
The advent of digital electronics allowing digital transmission has made TDM possible. Digital transmission is superior to analog transmission in several ways. It potentially has a very low error rate. Analog circuits have amplifiers that attempt to compensate for the attenuation in the line, but they can never compensate exactly for it, especially if the attenuation is different for different frequencies. Since the error is cumulative, long distance calls that go through many amplifiers are likely to suffer considerable distortion. Digital regenerators, in contrast, can restore the weakened incoming signal to its original value exactly, because the only possible values are 0 and 1. Digital regenerators do not suffer from cumulative errors. Also digital transmissions are possible at a much higher data rates, using existing lines. Digital Data services are now available in many cities, in the US. The Bell System has developed a digital network known as DDS (Dataphone Digital Service), it provides private point to point, full duplex, synchronous data rates at up to 1.544 Mbits/s. AT&T is offering DSDS (Dataphone Switched Digital Services), a switched facility at data rates of 56 Kbits/s. These services offer higher speed and higher reliability at lower cost than analog facilities. Digital transmission and its associated switching is likely to become much cheaper than analog transmission.

For data transmission there are basically two categories of TDMs. They are known as Byte-interleaved or Bit-interleaved TDMs. The first kind transmits one byte for terminal A followed by one byte from terminal B and so on, while the second kind transmits one bit from terminal A followed by one bit of terminal B and so on. Usually bit-interleaved TDMs are more appropriate for synchronous transmission, while byte-interleaved TDMs are more efficient for asynchronous transmission. Bit interleaved is totally transparent to data being transmitted over each channel and minimizes propagation delay through the system, thus permitting fast turnaround. Byte interleaved usually offers more flexibility and more efficiency for asynchronous transmission because it buffers a complete character before transmitting it, so it is possible to remove the start & stop bits.

Since the TDM is a digital system, it takes advantages of the higher speed of digital transmission. So not only does the TDM have a greater efficiency than the FDM, but the TDM's channel capacity is much greater than the FDM's as well.



a) TDM frame



[illegible]

[illegible]

host connection, make those technics inappropriate and not efficient. Indeed the big disavantage of TDM is that the time slot is allocated to the device whether the device has something to transmit or not. If there is no data to transmit in a given timeslot, a place holder or dummy character is used instead. It is not possible to skip a time slot, as the receiving end has no way of knowing that such a skip has taken place.

The application of microprocessor technology to multiplexing created the "intelligent" time division multiplexer, commonly known as the **Statistical Multiplexer** or Stat-Mux. Introduced in 1976, the stat-mux has ruled the multiplexer industry and more than 75% of multiplexers sold this year have stat-mux capabilities. The stat-mux is designed to handle the statistically determined data traffic more efficiently, due to its ability to store data temporarily in a buffer memory during periods of peak traffic.

In contrast to the conventional TDM which is driven by time, the stat-mux is driven by data. As a result of this feature, more low speed terminals can be attached to the stat-mux than can actually be serviced at any given time.

Figure III (b) shows an example of a stat-mux frame. Notice that the control field in the stat-mux frame is greater than the one in the FDM. But the characters in the data portion are real data as opposed to TDM frames that are filled with 'dummy' characters for non active devices. Unlike TDMs that do not require any protocol, stat-muxes need protocols to exchange data between the two multiplexers.

Protocols comparison

A protocol is defined as a set of rules for orderly interchange of informations between two or more devices.

Protocols are divided into two basic groups:

- Byte control oriented protocols (BCP)
- Bit oriented protocols (BOP)

Protocols provide many desirable features such as framing of data,

	BISYNC	DDCMP	X.25 LAPB
Protocol class	BCP Char. control.	BCP Char. count	BOP
Transmission format mode	Async/Sync Half duplex	Async/Sync Full/half dupl.	Sync Full duplex
Framing - start	2 SYNs	2 SYNs	Flag
- end	Terminating char	Char. count	Flag
Link control	Control char. optional header	Header required	Control field
Error checking	Info only	Header & info	entire frame
Error detection	VCR/LRC CRC-16	CRC-16	CRC-CCITT
request for retransmission	Stop & wait	Back n frames	Back n frames, selected reject
Max number of oustanding frame	1	255	7 or 127
Flow control	Control char.	None	Window scheme
Transparency	Transparent mode	Inherent (bytes count)	Inherent (bit stuffing)

Fig. IV: Common protocol characteristics.

Asynchronous transmission is fine when used on a device driven by an operator because the transmission is irregular and the start/stop bit attached to each character provides for the synchronization efficiently. But synchronous transmission makes far better use of the transmission facility by eliminating the start and stop bit. Furthermore, synchronous modems offer higher transmission speeds.

Full duplex transmission mode is more efficient than half duplex, because in full duplex operation, information can travel in both directions simultaneously between the locations. In half duplex, data flows only in one direction at a time, and the line must be turned around to reverse the direction of transmission. A 4-wire circuit, as opposed to a 2-wire circuit, is more efficient because it eliminates line turnaround which are time consuming.

Protocols that allow 'back n frames' type of retransmission are more efficient because frames do not need to be acknowledged on a one-to-one basis and may be outstanding (unacknowledged, unreceived or being received). It greatly reduces the number of turnarounds in half duplex and the number of control messages (i.e.: overhead) in full duplex transmission mode.

While some stat-muxes manufacturers are using proprietary protocols, Hewlett-Packard has chosen to use standard, broadly available protocols, at the Data Link Control level (referred as level 2 in the OSI model by ISO) for its line of Cluster Controllers, HP233x.

Why has Hewlett-Packard chosen Bisync, for its first product of the family, the HP2333A ? As can be seen from the chart above, Bisync does not seem to offer all features of some of the Bit Oriented Protocol. But, on the other hand, Bisync or more precisely the HP-multipoint protocol was readily available and already used to link terminals to HP hosts. So the integration of a new device like the multipoint cluster controller, on the existing product was a lot easier and could be achieved sooner. In doing that Hewlett-Packard has not only shortened the development cycle, but also provided a multipoint capability at no extra cost and minimized the hardware equipment involved by integrating the box on the system side into the system itself.

The Bisync protocol provides a reliable connection between the HP3000 system and the HP2333A multipoint cluster controller. The CRC-16 ensures error detection. The protocol also permits flow control of data between the host and the remote cluster controller.

Stat-muxes versus Cluster Controllers

Stat-muxes work in pair, one at each end of the link, while cluster controllers are analogous to terminal concentrators. They both use the same datacommunication concept of line sharing, but cluster controllers are supported by the host system and may not require special hardware on the system side, which is the case for the HP2333A, as it does not require any box sitting in the computer room. The advantage there, beside hardware cost saving, is the complete integration of the cluster controller to the host system (an HP3000 system for the HP2333A).

For example, stat-muxes will never be able to control echo from the remote box, the box at the terminal side, as the remote box has no way of knowing when to switch echo on or off. However, the MTS subsystem that replaces the box on the system side, knows exactly when to enable or disable echo because it is part of the host operating system. By using a high level protocol, above Bisync, MTS could at any point in time download that data into the remote cluster, thus dynamically altering the behavior of the cluster controller as requested by the application package. Not only the terminal reacts as if it was connected locally, but also it saves bandwidth on the line because the characters do not need to fly back and forth between the two boxes to be actually echoed by the host system.

MTS also, could concatenate write data coming out from the host so as to minimize the number of blocks transmitted on the line, thus increasing the efficiency of the utilization of the line bandwidth.

So, at any moment, the HP2333A knows exactly what it is supposed to do. It knows about echo, about break detection, about flow control between the device and the system (Xon/Xoff, Enq/Ack, Etc ...) and about read request characteristics issued by an application program or an HP subsystem. This ensures that data sent by the device is not split into two separate blocks interleaved with other blocks from other devices, thus leading the host to make wrong assumptions of what is happening. This phenomenon, well known by stat-muxes manufacturers, and usually called 'timing discrepencies', has forced most of them to create special options to alleviate those problems. Also by knowing the read characteristics, the cluster controller could concatenate data in order to minimize the traffic on the link. It also permits the Xon/Xoff or Enq/Ack protocol to be performed locally and to not transmit those characters on the link.

This logical (as opposed to physical) integration of the cluster controller into the system is the foundation of the 'transparent connection' concept. It permits a friendlier user interface because the cluster controller will react instantly to break signal or Xoff commands, giving the user an accurate feedback.

This integration also allows remote spooled printers to be supported. With the dual host option of the HP2333A, two different hosts (HP3000) could share the same remote printer. Outputs from each host are sent sequencially, one host waiting while the other one is using the printer. As a result, data from the two hosts are not mixed together, but come in orderly succession.

Stat-muxes architecture

Stat-muxes are built around one or several microprocessors. Figure V shows a logical block diagram of a stat-mux.

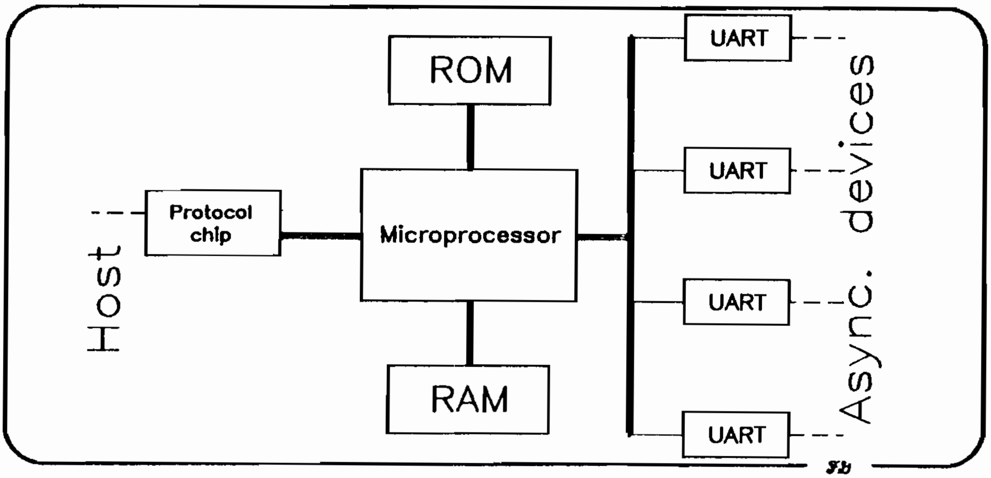


Fig. V: Stat-mux logical block diagram.

The asynchronous devices are connected through a UART (Universal Asynchronous Receiver Transmitter) LSI chip. It translates the serial bit stream into a parallel data path that interfaces to the microprocessor data bus. It also strips off the start/stop bits. Usually UARTs have programmable features such as, changeable line speed (from 50 to 19200 bps), adjustable character size (5 to 8 bits), parity generation and checking (odd, even, none), break generation and detection, overrun and underrun detection as well as framing error detection.

The protocol chip usually provides Sync or Async transmission, different line speeds, bit and byte synchronization, and some of the basic protocol functions such as bit stuffing for bit oriented protocol. Some protocol chips will do much more and take care of every aspect of the protocol up to level 2 or 3 (levels defined in the OSI model by ISO).

As can be seen from above, the microprocessor is pretty much freed of all those details and could be employed to perform other task such as optimizing the link utilization which remains the main resource in any multiplexing technic. Only the intelligence of a processor could actually obey an algorithm that, given the statistical nature of the data, will manage this resource efficiently. The processor buffers frames for each device using dynamic memory allocation scheme. Frames, ready for transmission, are inserted in the outgoing queue, transmitted according to the protocol, and retained in memory until a positive acknowledgment has been received from the other end. The same algorithm applies for incoming data.

At the same time, so as to not run out of buffer memory, the processor uses a flow control mechanism. Protocols provide flow control, so it is relatively easy for the multiplexer to protect itself against the other stat-mux at the other end of the link. By using well accepted flow control mechanisms such as XON/XOFF or CTS RS232 control line between the stat-mux and the asynchronous device, the stat-mux protects itself from overflow caused by external devices (terminals or hosts).

On the other hand, printers or CRTs are not always able to process or accept data at the datacomm speed, so stat-muxes must offer some features, that prevent those devices from being overflowed, such as XON/XOFF, CTS control line or ENQ/ACK.

Not shown on the block diagram are some miscellaneous components like: an EAROM or a CMOS-RAM with battery to retain configuration data during power failures. Also found sometimes is some added circuitry to help self-tests and self-diagnostics. Most of the time, a limited input and output peripheral is provided, ranging from a few DIP switches with LEDs to small tape cartridges and 7 segments digit LED.

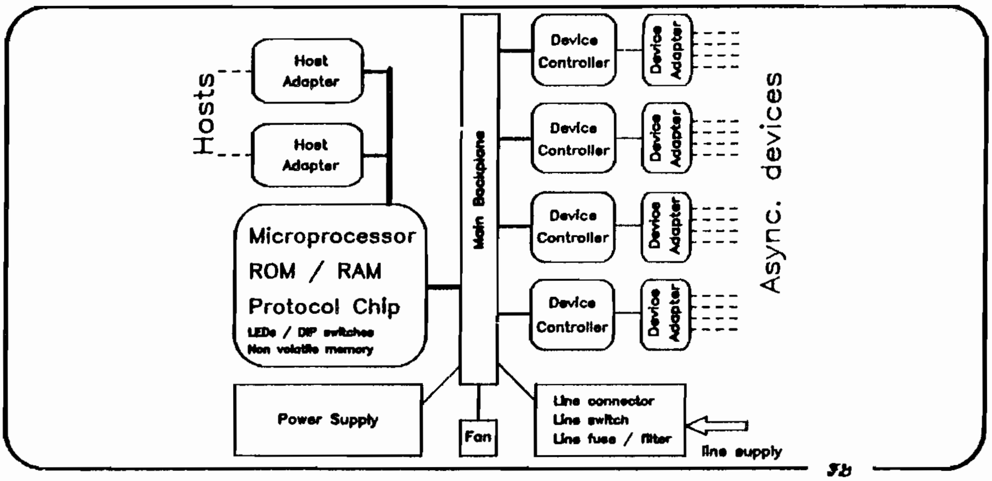

Hosts
Host Adapter
Host Adapter
Microprocessor
ROM / RAM
Protocol Chip
LEDs / DIP switches
Non volatile memory
Main Backplane
Device Controller
Device Adapter
Async. devices
Power Supply
Fan
Line connector
Line switch
Line fuse / filter
line supply
FB

Fig. VI: HP233x architecture.

The HP233x Cluster Controller architecture is very close to the above description, but for added flexibility and power, a microprocessor has been added for every four UARTs, so protocol between the cluster controller and the Asynch devices will not interfere with anything else. As seen on figure VI, the design is extremely modular, thus allowing room for future enhancements. Adapter cards are provided for the implementation of level 1 (as per ISO), allowing the flexibility to add any new electrical interfaces as needed by the market.

Stat-muxes features & benefits

Stat-muxes improve link utilization. For asynchronous workstations, the stat-mux takes advantage of the bursty nature of data-transmission and is far more efficient than TDM or FDM technics. Depending on data traffic, stat-muxes usually accommodate three or four times as many devices as TDM, while maintaining an acceptable performance.

Performance of stat-muxes is not easy to determine. Performance indicates how efficiently the stat-mux can transfer data from one end to the corresponding port at the other end of the link. But depending on the internal multiplexing method, stat-muxes may be more efficient than others, upon heavy traffic condition or error prone link. To increase link utilization even further, some stat-muxes offer data compression. Efficiencies of 75% to 90% are attained with these technics when used in certain applications. However data compression algorithms are very sensitive to the data stream and should be used with caution after thorough analysis of the data transmitted.

The main reason for using stat-muxes remains cost objective. A voice grade channel offering services at up to 9600 bps is not much expensive than a channel limited to a lower speed. Thus if more data can be added to one channel and some slower channels eliminated, there is an excellent prospect for trading off reduced line costs against the cost of multiplexing equipment.

Costs involve the monthly charges for the line and the purchase price of modems and multiplexer. When comparing the different alternatives, detailled cost analysis demonstrates that for two terminals located at more than 50 Km of the host system, savings are realized within the first year of usage of multiplexing equipment. Cost analysies are heavily dependant on line cost and may vary considerably from one country to an other. As the number of remote devices increases, the saving becomes higher and the hardware cost of multiplexing equipment is paid for even sooner.

But stat-muxes offer more than cost savings. They also provide reliable transmission. Asynchronous terminals do not provide for error free transmission because they usually do not use protocol to exchange information with the host. Stat-muxes, with their protocol guarantee virtually error free data transmission between the host and the workstation by retransmitting erroneous data. Also newer stat-muxes offer extended diagnostics capabilities, which increase even further the reliability of the installation. Diagnostics help troubleshouting the installation. They offer tests that could be activated for one terminal, on line, without disrupting operation of others devices. They provide local and remote loop back that permit the operation of every equipment of the network to be tested easily. Some units also indicates link utilization or internal buffer memory allocation. Those indications are very valuable for the data communication manager because they tell him when it is time to buy some new pieces of equipment, change the link's data rate or balance the traffic in different way.

These functions are the basis of an evolution of stat-muxes toward networking devices. With these added features, stat-muxes are coming to play an important role in networking. These networking functions are more powerfull, easier and friendlier to use through the monitor port or supervisory port. Since the supervisory port is no longer used as a data channel it can support interactive devices, and can be used to monitor the actual data traffic on a given data channel, output statistics, configure link protocol characteristics as well as data channel parameters and initiate diagnostics.

The switching and port contention capabilities are the next step toward a full networking device. Using the connection process of the switching capability, the user can choose, from each individual devices, the host system he wants to connect to. On the other hand, the port contention feature permits better usage of the resources because not only does the aggregate speed of all devices exceed the available link data speed, as with conventional stat-muxes, but since not all devices are 'logically' connected continuously, the number of devices 'physically' connected to the network is increased allowing potentially more users to be served. At the same time it diminishes the number of ports used on the host thus saving on the total datacommunication cost.

HP233x Cluster Controller family

The main objective of this product line is to provide cost effective, reliable and transparent connections of workstations to HP host systems. Neither the user at the terminal, nor the application program controlling this terminal could 'see' the cluster controller. It works as if the terminal or the printer were connected locally to the system. It thus benefits from new datacommunication technology and at the same time uses the existing installed base of terminals as well as newer ones.

There are three different protocols used to connect an HP cluster controller to an HP host system:

- HP-Multipoint Bisync
- X.25
- IEEE 802

The first model of the family, the HP2333A multipoint cluster controller, was introduced in late 1982. This model is only supported on the HP3000 series. It supports up to 16 workstations; terminals, personnal computers or printers. It has two electrical interfaces to the host, in order to cover both local and remote environment. In local applications, it is recommended to use the multidrop DSN/Data-Link. The Data-Link is a single shielded twisted pair of wires, extending for a length of up to 4 Km which permits connection of one or more HP2333A units as well as individual terminals. For remote applications, an RS-232 interface connects the HP2333A to the modem. The optional second host interface, gives the HP2333A a limited switching capability. In that case each device can independantly select one of the two hosts, or the hosts could share the same printer attached to the HP2333A using remote spooling capability.

Figure VII shows a HP2333A connected locally to host A through a Data-Link, and connected through a modem to host B. Unlike regular stat-muxes, only one box, on the terminal side is required. The MTS subsystem with its associated hardware, the INP controller, performs the task usually performed by the box on the system side. This permits a transparent connection. (As explained previously in details in 'stat-muxes versus cluster controllers').

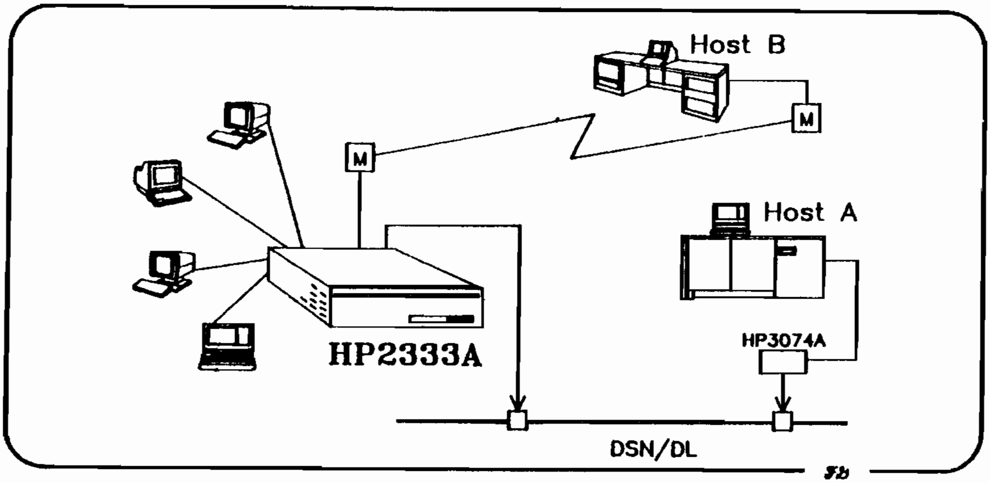


Fig. VII: HP2333A configuration.

Introduced in the fall of 1983, the second model of the family, the HP2334A uses X.25 protocol. It provides for remote connections through Public Data Networks (PDN). A typical HP2334A configuration, shown in figure VIII, illustrates the flexibilty of PDN. Each terminal connected to the HP2334A could access any host, or in others words, have access to any resources of the network. The PDN provides the switching capability and it is a very cost effective solution when terminals need to access many hosts geographically dispersed. The HP2334A could also be used to permit a terminal access, through PDN, to a host that does not support X.25 protocol.

As an enhancement to this offering, an updated HP-Host-X.25 software, that will permit transparent operation of workstations through PDN with HP equipment, is currently under development. Upon introduction of this upgrade, most user written application packages, or HP subsystems, will provide the same functionality with terminals or printers connected remotely on an HP2334A through PDN as they do locally.

To be introduced, shortly after, is a true stat-mux configuration using HP2334 technology with one box at each end of the link. Those boxes will be able to communicate through a PDN or directly using modems on leased line.

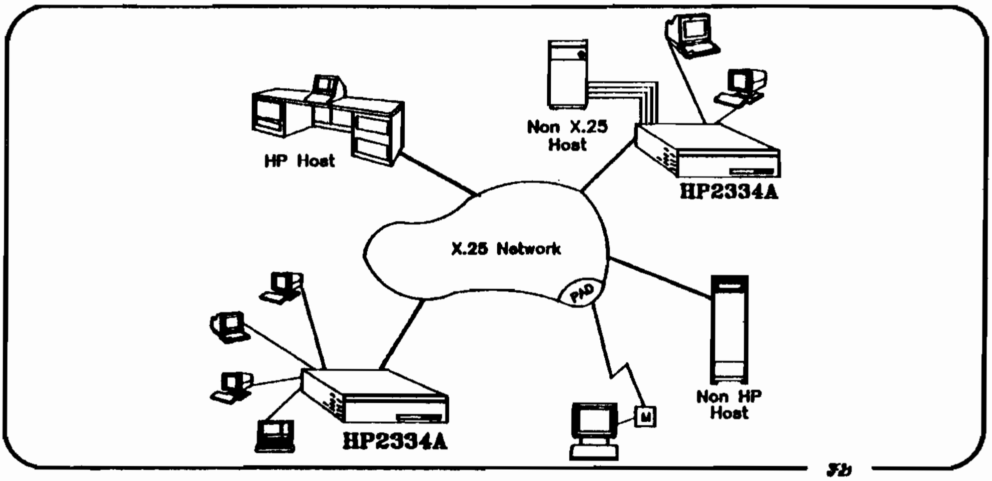


Fig. VIII: HP2334A configuration.

The next model of HP cluster controllers will provide transparent connection of workstations, through Local Area Network (LAN). It will uses the IEEE-802 protocol. The LAN will provide the switching function, so any of the 32 workstations connected to a IEEE-802 cluster controller will be able to access any host on the network. When using HP equipment, workstations and hosts, the connection will not be visible. Application packages will continue to operate the devices as if they were connected directly on the asynchronous multiplexer of the host.

The HP233x cluster controller product line offers a **supported** and **transparent** connection of HP workstations to HP hosts. While the performance of line sharing may never be as good as direct point-to-point connection, multiplexing of devices over a single line provides tremendous cost savings in remote environments (reduces modem and telephone line costs) and easier yet cost effective solutions for local environments. As the number of workstations increases, cluster controllers will be THE way to interconnect easily that many devices with a good cost/performance ratio.

Criteria For Selecting Software Packages For The HP3000

By Thomas A. Wilson
Director, PROFILES/3000 Sales
COMSHARE, Inc.

Criteria For Selecting Software Packages For The HP3000

By Thomas A. Wilson

It used to be that you would buy some software, load it up, pay the vendor and then the real work for MIS would begin. With the common reductions in the price of hardware, there is more pressure on the software suppliers to differentiate themselves from each other. This means opportunity for the MIS manager in charge of purchasing software. Suddenly phrases like, "Let's try it out; Show me; Can you prove that?" and "Tell me about your support after the sale" have real meaning and can play a large role in the software that you ultimately select.

I feel that there are 10 objective decisions for your consideration. Having completed that task you will have one subjective decision yet to make.

The following information if properly employed should lead you to a good solid selection that will have an ongoing positive impact on your productivity over the life of your selected software.

I. Documentation -- Everyone talks about documentation and fortunately everyone is forced to do something about it. Unfortunately the normal function is to do as little about it as necessary. No documentation is going to fit specifically within your own standards, but some common questions that have been asked are:

1. Does the documentation reside on-line for quick and easy accessability?
2. Can I understand the documentation?
3. How many kinds of documentation do you have? i.e., technical, user, training.
4. May I see the user documentation?
5. What does the training documentation look like? (This is in case I need to give additional training classes in the future.)
6. What does the technical documentation look like?
7. It is understandable?
8. Is it consistant with any of my other documentation?
9. Can it be changed and if so, how easily?

As you can see there are many considerations from just one decision criteria. Without a specific plan it is possible to make what appears to be a good decision but, find out later that it was an incomplete decision that led to disaster.

II. Training -- This is a task that quite often falls on the shoulders of MIS. The ideal situation would be a training class given by the vendor with enough documentation that additional classes can be taught by a representative from the user organization. Some key questions for your consideration include:

1. Is the training conducted on site?
2. Is it classroom or terminal session oriented? (Tests show 400 percent better retention with terminal session classes.)

3. Is the training documented for easy reference and also for ongoing internal training? Is this documentation "user friendly" so that the layman can teach the class?

If your current policy is to not give internal training classes, you need to know if additional training classes are available and if so, what are the specific terms? Will the company come back to you or will you need to go to them? What are the charges for additional training and how long are those prices good? Is there a guaranteed response time for additional necessary training?

Training is a very important aspect in your decision because the software decision you make will be openly evaluated by the end user and judged on his/her ability to make the system do what they want. With incomplete or poor training an excellent piece of software will be given a miserable rating, and guess who will be blamed for this bad decision?

III. Service and Support -- This area is probably the one that is undergoing the greatest amount of change in the shortest period of time. It is an area that is very costly for you and one where the most vendor differentiation is taking place. The functions to be dealt with include hot lines, trainer follow-up, bug requests, and product enhancement input.

1. The most controversial consideration is the utilization of a hot line service. It raises the basic question of control, but properly implemented, can be controlled by MIS with a very high degree of increased productivity. End user problems have historically been the problem of MIS. Countless hours have been wasted researching non-problems. Eighty-five percent of a user's problems in the first six months are education and not system oriented. The good vendor will want to respond to these questions because he or she can spot trends and recognize necessary changes for future training classes. Also, patterns of questions in a multiple client relationship help easier and more efficient user instructions.
2. All software contains bug request facilities in the maintenance agreement and this is an area to verify.
3. Enhancement input is a method of increasing your productivity. Find out the process undertaken by your proposed vendor and determine how much impact you may or may not have in this decision process. Some companies welcome your suggestions and given that they have a multiplier effect will implement them as maintenance. Others have internal departments that are not interested in outside input. Ask your vendor's policy and then ask for proof of this policy by verifying implemented changes and talking to references.

This entire process will be a potential area of increased productivity over the next few years. It is one that is now highly people intensive and requires a high level of end user knowledge. In all probability, this is a luxury that you will not be able to afford in the near future.

IV. Equipment Requirements -- This is an area that you are probably most comfortable in but one that has a few pitfalls. Just because the software has an Image data base doesn't mean that it will fit your needs. You may be interested in whether it was designed specifically for the HP 3000 or perhaps it was converted from an IBM, Burroughs or some other system. You need to look at the design from an efficiency point of view. A natural question is, "What are the minimum machine requirements and can I benchmark its productivity?"

V. Installation Ease -- After determining the efficiency of the software it stands to reason that you might be interested in how much work is involved in installing the package. No package is going to meet all of your musts and wants, so I would be especially curious as to the difficulty involved in modifying what I was buying and also what impact if any, that it had on the validity of my maintenance agreement. Additional information that will have direct impact on the success of the installation include:

1. Must all data be input initially in order for the system to function?
2. Is a batch load facility available to initially load the data or must it be loaded item by item?
3. Are there rules established to insure that the data is valid or must all data be visually verified?

Another important aspect of the installation is the ability to identify, measure, and control the entire process. Ask for proofs of timeframes and controls, discuss the company's claims in this area with some of the installed users.

VI. HP Interface -- It is important to many companies that there is consistency in their software so the question of HP interface is important. The questions that readily come to mind are:

1. Will this package interface with Rapid, Inform, Query and also, what about purchased packages like Quick and Quiz?
2. Can I interface with word processors?
3. If I hook up on HP micro, can I also use it as a terminal with this software?
4. What are the requirements and limitations of terminal selection with this package?

VII. Maintenance -- What do I get and what does it cost? Not bad things to determine early on in your selection process. As was mentioned earlier, the phone support issue is critical and becoming more so as people costs go up. What is the company's policy around fixing bugs, and if they are fixed do I get both source and object code on the fix? What about ongoing enhancements? Am I going to get ongoing improvements or am I caught up on the nickel and dime treadmill of continuous additional modules? This specification can have an impact on the true actual cost of your system.

Ask the vendor to show you proof of maintenance. This should be easy to do and will satisfy any apprehension you may have surrounding maintenance as an issue.

VIII. Security -- All systems will have some form of password security and all security can be broken. Most end users will not have the ability to break any security controls so my suggestion is to check additional security features and their associated benefits. Things to look for include levels of password security and the ability to institute security to the data item level it desired. One benefit of higher levels of security is that it can be controlled by the account supervisor and changed without having to involve MIS to unload and reload the data base.

IX. End User Understanding -- As more end users start using the computer, communication becomes critical. The terminology used by end users can be as frightening to the MIS department as the data processing terminology is to the end user. It is for those reasons that you want to determine not only the technical level of expertise of your preferred vendor, but also their understanding of the ultimate users' needs. This should include not only knowing what and why the system needs to have certain features but also what will the needs be in the future. Has the system been developed with those future needs in mind and considerations made to accommodate those enhancements? If this vendor has passed your preliminary screens, it won't take long with the end user to see if they can communicate effectively.

It would not be out of line for you to ask if verification of this company's credentials is the expertise desired by your end user.

X. Vendor Viability -- This is the area that in many cases is the only area scrutinized. I contend that the important aspects of this area are as follows:

1. How old is the company you are dealing with?
2. What is the financial stability of the company I am dealing with? (This is critical in today's economy. What happens if the company goes out of business and what is the potential probability of that happening?)
3. What is the geographic presence of this vendor? Do they have 10 people or a thousand, and are they all located in one geographic area or are they accessible in many locations?
4. What is the company's reputation? Do they follow through on commitments? What about other product lines? Can I verify the company's commitment to my needs?
5. Will the company steer me to one or two special references or can I get a list of clients and talk to whomever I wish?
6. How long has the company dealt in the functional area of my needs? How much do they know about my business and my end user's needs?
7. How long have they dealt in my specific marketplace? Do they understand my business? How well do they know the HP 3000? Am I a small part of their software service and controlled by what IBM users want or am I afforded equal treatment and consideration as an HP user?

XI. The Subjective Emotional Decision -- If two or three vendors happen to pass your screens and only minor criteria differentiate them, chances are you will choose the company whose representative has gained your confidence. As a matter of fact, most software selection is made based on the relationship and amount of trust built up between the sales person and the decision maker or chief recommender. This is not bad decision making on your part, particularly if you are aware of it before making your final decision. As in all business decisions, trust is a major consideration. That being the case, accept it and use it to your advantage.

I truly believe that if you consciously consider all of the factors listed above that you will do a better job of selecting software for your organization and that you will enhance your potential for advancement by measureably increasing productivity.

PLANNING AND IMPLEMENTING A CORPORATE DATA CENTER IN AN INTERNATIONAL ENVIRONMENT

Norman W. Davis
Teleco Oilfield Services Inc.

Introduction

There are many problems and complex alternatives in developing and installing a world-wide computer network. This paper discusses the problems and solutions of a rapidly growing multinational company that has central data processing control, and supports multiple computer sites of various sizes throughout the world.

Background

Teleco Oilfield Services, Inc. is a high technology company that provides measurement-while-drilling (MWD) directional drilling data services, primarily to drillers of offshore oil and gas wells. Teleco is engaged in research and development programs to develop enhanced MWD systems designed to provide in addition to directional drilling data, other data for well control, identification of geological structures and improving drilling operations.

Teleco's directional MWD system includes a sensor assembly and transmitter mounted in a non-magnetic drill collar, and a data display unit located on the drilling platform. The Teleco downhole tool takes measurements pertaining to the direction of the well, and orientation of tools used to control the well path, and it sends those measurements to the surface by inducing coded pressure pulses in the flowing mud column inside the drill pipe. Teleco's surface unit decodes the pulses and displays the measurements. This technique reduces substantially the total time required to take directional measurements, compared to non-MWD methods now commercially in use, and permits an increase in drilling efficiency by allowing the driller to maintain a higher daily drilling rate and better control of the well path through more frequent measurements. By permitting continuous mud circulation and shortening the time the drill string is not rotating the system also reduces the risk that the drill string will become stuck in the well.

This directional drilling procedure is highly successful, and Teleco's MWD service is the most extensively used system in the world, currently having provided service to more than 1400 wells worldwide. In 1979 Teleco became a public company and completed its first full year of commercial operations. Today, Teleco's revenues exceed 60 million dollars, it currently has over 700 employees, and has major operations in the United States and the United Kingdom, and additional operational facilities in West Africa, The Middle East, and the Far East. The Home Office is located in Meriden, CT, with two regional offices, located in Houston, Texas and in Versailles, France.

Computer Requirements

Teleco's rapid growth and worldwide organization presented major problems for a data processing organization. The company had a variety of financial and operational data originating from remote locations throughout the world, with the requirement of processing the data first in the regions and then transmitting the data to the home office in the United States for final processing. To facilitate information gathering, remote processing and transmission of data, the requirements for computer hardware varied significantly.

The home office in Meriden, CT in the United States required a large central processing unit capable of supporting multiple users, many applications, and major software development efforts. The regional offices in Houston, TX and in Versailles, France required a smaller central processing unit and a smaller number of peripherals, yet the software developed on the larger system in CT had to be totally compatible. Teleco also has two large field maintenance depots, (FMD) located in Lafayette, Louisiana and Aberdeen, Scotland which also required computer capability but less than that required in the regional offices. The lowest level of computer requirement was for the remote locations throughout other parts of the world that needed data processing support on a local basis, and yet must have system compatibility and the ability to communicate to the regional and home offices. (See Figure 1)

The Approach

To meet these requirements Teleco organized a department called Corporate Information Systems and assigned it the responsibility for all data processing on a worldwide basis. The objective of Corporate Information Systems was to meet Teleco's worldwide information requirements with accurate, timely, and cost-effective systems.

Because Teleco was a new company, it had not automated any of its general business applications, and at that time, only had a manufacturing system operating on an IBM System 34. Corporate Information Systems determined that the Teleco environment, which was experiencing rapid growth, expanding worldwide locations, and identifying significant information requirements, presented major dangers for a data processing organization. These dangers included excessive and redundant costs, independent and incompatible systems, lack of standardization, loss of accountability, poor data security, and uncontrolled growth of incompatible hardware and software.

CIS determined that a centralized approach was required in order to avoid these dangers and to best meet the long range objectives of the corporation. The advantages of central control were:

1. Centralized Systems Development - All system analysts and programmers would be located at the corporate office.

2. Corporate Wide Standardization - All standards would be established by the corporate group and then sent to the field.

3. Maximum Data Security - Data file access would be controlled centrally, one system manager, and centrally developed system control.

4. Proper Financial Controls - Corporate developed and maintained audit trials.

5. Ease of Systems' Maintenance - Bug fixes, changes and enhancements made centrally then sent to the field. One source code location.

6. Rapid Information Retrieval - Each organizational level has complete data files of lower levels for immediate reporting.

7. Maximum Cost-Effectiveness - Single development staff, standard hardware and software, single site system support, and reduced management positions.

In the summer of 1982 CIS undertook the task of developing and implementing a world-wide computer operation. Three major objectives were identified:

1. Conduct a survey of the total corporate computer application requirements.
2. Select and install the proper hardware and software to meet these requirements.
3. Build an organization that could develop, implement and maintain these computer systems.

Application Survey

All user departments were surveyed to identify their application requirements and approximately fifty-five unique requirements were documented.

In order to properly evaluate and assign priorities to these applications, a data processing Steering Committee was formed, which was made up of major department managers who would meet on a periodic basis and review and evaluate the individual applications submitted to them. Key criteria were developed, based on the company's long range plans and strategy, to be used by the Committee for evaluating each application. All the applications were reviewed and evaluated by the Committee, priorities were assigned, and a detailed development schedule was made resulting from the priority assignments.

Hardware Selection

During the application survey and application evaluation process, the hardware selection process was in progress. The company's IBM S/34 was providing limited data processing support to the organization and was not capable of meeting the future requirements. Therefore, it was determined that a completely new computer system plan to meet world wide requirements had to be developed.

A request for proposal (RFP) was developed and sent to 9 computer vendors. The RFP required the vendor to submit a proposal in a fixed format meeting specific requirements concerning hardware, software, cost, upward growth capability, peripherals, documentation, delivery time, benchmark test results, and a variety of other criteria (See Figure 2). When the proposals were received from the vendors, original screening and evaluation were completed and the vendors were narrowed down to five finalists. These five were invited in for oral presentations and a question and answer session before final selection was made. The five finalists were Burroughs, Data General, Hewlett Packard, IBM, and Prime.

Corporate Information Systems, considering the company's total requirements and needs, developed 24 evaluation criteria to be used to evaluate the individual hardware vendor's proposals. (See Figure 3)

Each one of the evaluation criteria was weighted from 1 to 15 points, and the highest possible score that a vendor could acheive was 1960. The criteria were weighted based upon their importance to and impact on Teleco's objectives. Hewlett Packard received a score of 1864.4 with a 95% rating. Prime was second with an 80% rating, and IBM and Date General were third and fourth. (See Figure 4) In terms of price, Hewlett packard had the lowest price proposal of the five vendors. It is important to point out that the RFP required each of the vendors to propose and price the identical hardware configuration, in other words, the same number of CPUs in the various locations, the same number of terminals, printers, plotters, tape

TELECO COMPUTER NETWORK

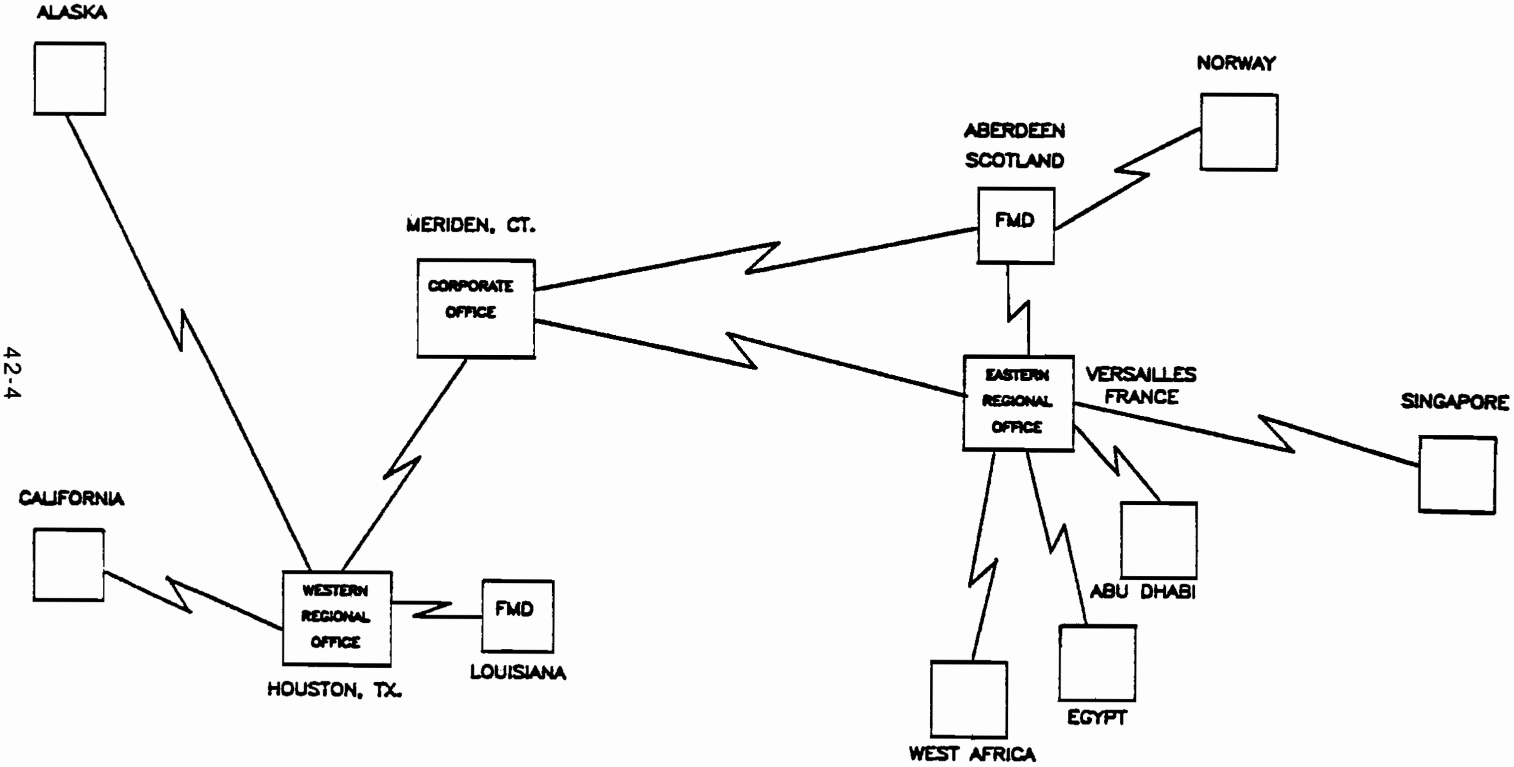

ALASKA
NORWAY
ABERDEEN
SCOTLAND
FMD
MERIDEN, CT.
CORPORATE
OFFICE
EASTERN
REGIONAL
OFFICE
VERSAILLES
FRANCE
SINGAPORE
CALIFORNIA
ABU DHABI
WESTERN
REGIONAL
OFFICE
FMD
HOUSTON, TX.
LOUISIANA
EGYPT
WEST AFRICA

Figure 1

REQUEST FOR PROPOSAL

TABLE OF CONTENTS

I. INTRODUCTION

II. DESCRIPTION OF TELECO

III. SYSTEM REQUIREMENTS

  A. Main Office
    1. Hardware
    2. Software
    3. Communications
  B. Regional Office
    1. Hardware
    2. Software
    3. Communications

IV. SELECTION PROCESS AND SCHEDULE

  A. Time Table
  B. Evaluation Criteria
  C. Definition of Benchmark Tests
    1. RPG II
    2. Cobol

V. REQUIRED PROPOSAL OUTLINE

  A. Cover Page
  B. Executive Summary
  C. Table of Contents
    1. Introduction
    2. System Description
      a. Hardware
      b. Software
      c. Communications

Figure 2

3. Cost Summary
    a. Corporate Office - Tables I A, I B, I C
    b. Regional Office - Tables II A, II B, II C
    c. Cost Summary - Table III
4. System Growth Potential
5. Implementation Plan
6. Benchmark Test Results
    a. RPG II
    b. Cobol
7. Applications Software
8. Vendor Support
9. Maintenance
10. Warranty and Trade-in Policy
11. Environmental Considerations
12. References

APPENDIX A - System Evaluation Questionnaire
APPENDIX B - Sample Contracts
APPENDIX C - Annual Report and Financial Information
APPENDIX D - Additional Information

ATTACHMENT I - Financial Statement and General Information
ATTACHMENT II - RPG II Source Listing
ATTACHMENT III - COBOL Source Listing

Figure 2 (cont.)

VENDOR EVALUATION

Score: Weighted / Base

EVALUATION CRITERIA	WEIGHT	BURROUGHS		DATA GENERAL		HEWLETT PACKARD		IBM		PRIME	
		Base	Weighted	Base	Weighted	Base	Weighted	Base	Weighted	Base	Weighted
HARDWARE COMPATIBILITY	16	8	120	6.5	97.5	10	150	5.4	81	8	120
SOFTWARE COMPATIBILITY	15	4.8	72	8.8	132	10	150	5.8	87	8.2	123
SYSTEM GROWTH CAPABILITY	14	6.6	78.4	9.6	134.4	10	140	5.6	78.4	9.2	128.8
DATA BASE CAPABILITY	13	7.4	96.2	7	91	10	130	9.6	124.8	6.8	88.4
COMMUNICATIONS CAPABILITY	13	10	130	10	130	10	130	10	130	10	130
FINANCIAL SOUNDNESS	12	6	72	8	96	9	108	10	120	7	84
WORLD WIDE SUPPORT	11	10	110	7.6	83.6	9	99	10	110	8.8	96.8
M & D GENERAL LEDGER PACKAGE	10	5.6	56	.4	4	10	100	9.6	96	6.6	66
COST — HARDWARE & SOFTWARE	10	2.8	28	8.4	84	10	100	5.6	56	7.2	72
PROGRAM DEVELOPMENT PACKAGES	9	7.6	68.4	8.2	73.8	10	90	9.4	84.6	8.4	75.6
MAINTENANCE SUPPORT	9	6.2	55.8	7	63	9.4	84.6	9.6	86.4	7	63
EQUIPMENT DELIVERY TIME	8	5	40	6.8	54.4	10	80	9	72	10	80
USER DOCUMENTATION	8	10	80	10	80	10	80	10	80	10	80
APPLICATION SOFTWARE	7	6.2	43.4	8.8	61.6	10	70	5.6	39.2	7.4	51.8
NON-EDP USER FRIENDLY	7	6.2	43.4	8.8	61.6	10	70	9	63	8.8	61.6
EDUCATION & TRAINING	6	8.2	49.2	8.6	51.6	9.4	56.4	8.2	49.2	8.2	49.2
FULL LINE OF PERIPHERALS	5	5.4	27	7.8	39	10	50	3.4	17	8.2	41
CONVERSION EFFORT & SUPPORT	5	6.4	32	7.2	36	8	40	9.8	49	7.6	38
ENGINEERING & SCIENTIFIC COMPATIBILITY	4	0	0	8.2	32.8	10	40	0	0	8	32
IMPLEMENTATION & PLAN SUPPORT	4	5.6	22.4	5.6	22.4	9.6	38.4	6.8	27.2	5.8	23.2
BENCHMARK TEST RESULTS	2	3.4	6.8	8.2	16.4	5.6	11.2	8.4	16.8	6.2	12.4
STAFFING IMPACT	2	4.2	8.4	9.4	18.8	9.6	19.2	10	20	9.4	18.8
ALTERNATE SITE BACKUP	2	8.8	17.6	8.4	16.8	9.6	19.2	9.4	18.8	8	16
SITE PREPARATION	1	3.4	3.4	10	10	8.4	8.4	10	10	9	9

Figure 3

COMPUTER SYSTEM

EVALUATION RESULTS

- Number of Evaluation Criteria 24
- Criteria Weighted from 1 to 15
- Highest Possible Score 1960

	Burroughs	Data General	Hewlett Packard	IBM	Prime
Score	1260.4	1490.7	1864.4	1518.4	1560.6
Percent	64%	76%	95%	77%	80%

All vendors were rated on all twenty-four criteria by all five members of the evaluation committee. (Appendix 1)

Figure 4

drives, etc.

In our opinion, one reason why Hewlett Packard's rating was so high, was their commitment to total compatibility throughout their product line. Teleco's organizational structure required small computer configurations in the remote locations, medium sized systems in the regional offices, and a large development system at the home office, yet we required total compatibility between all these systems. Hewlett Packard's abiity to have the same software operating on a small computer at the Feild Maintenance Depots, a medium size system at the regions, and yet have that same software developed and operate at the Home Office, gave them a significant advantage in meeting our needs. For example, IBM had to bid System 38s at all locations in order to meet the compatibility requirement. In addition, the upward growth capability which was a strong requirement, again allowed HP to score highly in this criteria.

After a visit to the Hewlett Packard corporate offices in California for discussions with management on their future plans and committments, HP was selected as the vendor and a contract was signed in July 1982, with hardware scheduled for delivery in September of the same year. I'm happy to report that HP delivered the home office configuration two weeks ahead of schedule, and all the additional sites throughout the world were delivered on schedule.

The complete computer configuration for each of the worldwide locations consisted of the following equipment: The Teleco Corporate Office in Meriden, CT U.S.A., consisted of an HP3000 Series 64 with three megabytes of memory, two 7933H 404 MB Disc Drives, one 2619A high-speed printer, a 7976A high-density tape drive, and approximately 65 terminals. Teleco has two regional offices, a Western Regional office in Houston, TX U.S.A.; and the Eastern regional office in Versailles, France. The Houston computer configuration consisted of an HP3000 Series 40, one megabyte of memory, one 7933H 404 MB Disc Drive, 2608S 400lpm printer, 7970E tape drive, and approximately 10 terminals. The Eastern Regional office in Versailles, consisted of the identical configuration as in the Western Regional office. Within each region, there is a field maintenance depot that also required on site computer processing. In the Western Region this field maintenance depot is located in Lafayette, Louisiana. Because of the proximity of Louisiana to Texas and the reliable telephone communications between the two locations, we were able to avoid the expense of an additional CPU, and made this a remote site connected to the Western Regional office. A dedicated lease line was established between Houston and Lafayette, and installed at this facility were 6 CRTs and a 2608S medium speed printer.

In the Eastern Region the field maintenance depot is located in Aberdeen, Scotland. Because it would require crossing country boundaries, we felt it would be difficult to have a lease line between the Aberdeen facility and the Versailles regional office. For this reason we decided to have a separate computer configuration in Aberdeen. This configuration consisted of an HP3000 Series 30R, which was a refurbished machine yet fully warranted and maintained by Hewlett Packard. This system had close to the same capabilities as the systems located in the regional offices, but it was significantly more cost-effective.

The CIS Organization

During the hardware selection and evaluation process, the CIS organization was being fully staffed. The organization chart is shown in Figure 5. Teleco had an immediate need to get the worldwide communications network operational as quickly as

TELECO
CORPORATE INFORMATION SYSTEMS

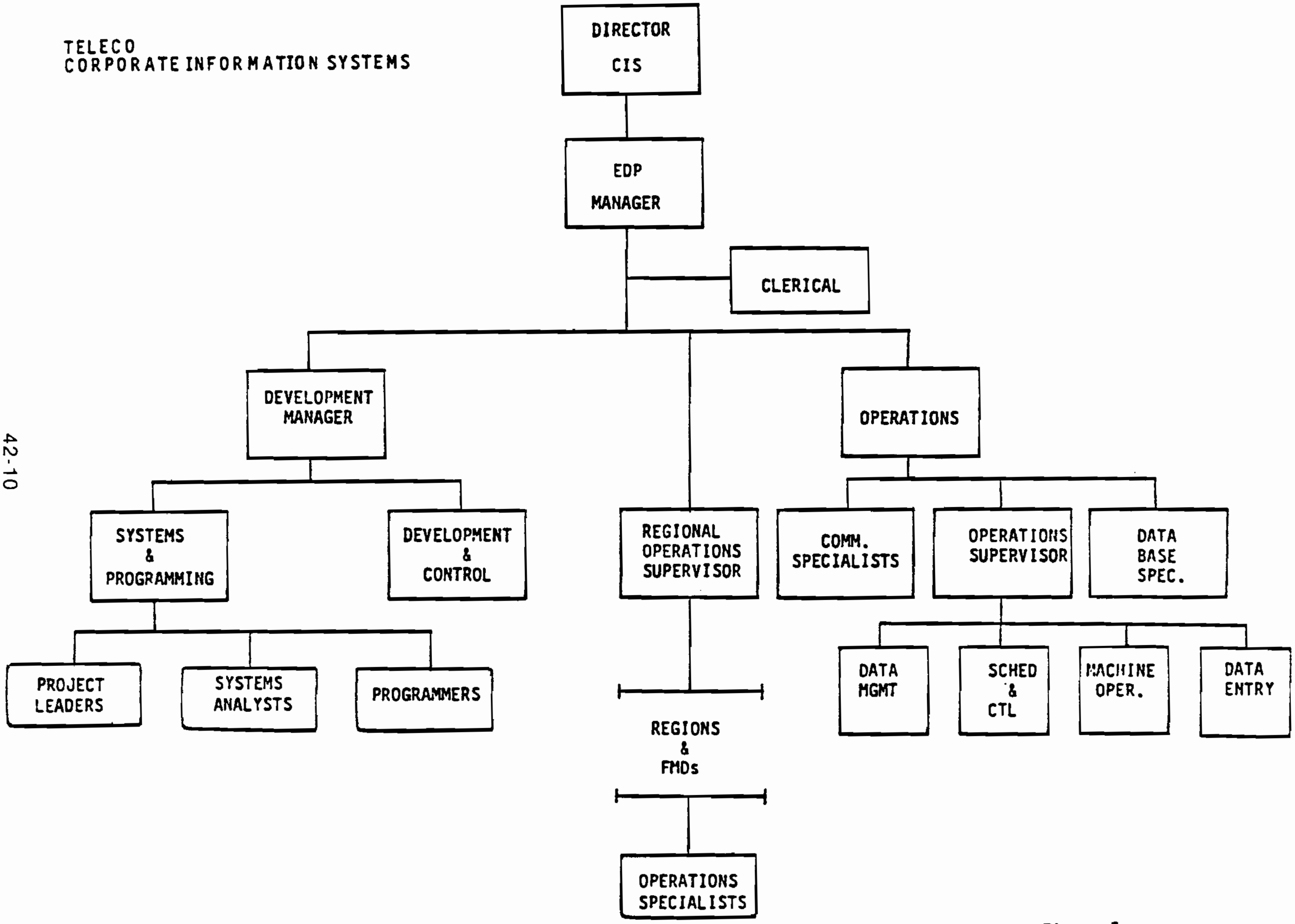

DIRECTOR
CIS
EDP
MANAGER
CLERICAL
DEVELOPMENT
MANAGER
OPERATIONS
SYSTEMS
&
PROGRAMMING
DEVELOPMENT
&
CONTROL
REGIONAL
OPERATIONS
SUPERVISOR
COMM.
SPECIALISTS
OPERATIONS
SUPERVISOR
DATA
BASE
SPEC.
PROJECT
LEADERS
SYSTEMS
ANALYSTS
PROGRAMMERS
REGIONS
&
FMDs
OPERATIONS
SPECIALISTS
DATA
MGMT
SCHED
&
CTL
MACHINE
OPER.
DATA
ENTRY

Figure 5

possible in conjunction with the field applications.

The critical positions that were filled first were the Development Manager, Operations Supervisor and Regional Operation Supervisor. We recognized that the key to successful regional computer operations that are controlled centrally was insuring that the remote locations were fully and properly represented at the home office. This was accomplished by the Regional Operations Supervisor position. The major functions of this position are to record the needs of regions, to represent the regions at the Steering Committee, to communicate worldwide all corporate policy, procedures and new software releases, and provide functional management over all remote computer locations.

The Data Base Administration and Communication Specialists were the other key positions that played an important role in the development and implementation of our worldwide computer network.

Whenever we could, we hired individuals with previous Hewlett Packard experience. However, we found that a good programmer with IBM or other vendor experience could become productive in a short period of time after attending the HP introduction courses and supplemented with some of the HP self-paced instruction packages.

Software Development Policy

The important factor in our computer network was the requirement to have the same software that was developed on the Series 64 in Connecticut be totally compatible at all computer configurations throughout the Teleco operation. A software change made to an application program at the Home Office could be communicated directly to the other CPUs throughout the network and immediately be operational on those systems.

All systems development work, programming, and testing are done at the corporate offices at Meriden, CT. There are no systems people, programmers, or analysts located at any of the other facilities. It is a requirement that all application software be compatible throughout the corporation, regardless of the requirements at some of the remote locations, whether it's in the Eastern Region or the Western Region, total compatibility is the theme throughout all application development. We have insisted that if a single change is made to an application program, it will be made once at the Home Office and then distributed throughout the network. Enhancements, modifications, and bug corrections are also made at the Home Office, tested and then transmitted back to the remote locations.

All systems management functions for the Western Regional office are handled by the systems manager located in the Corporate Office in Meriden, CT. However, because of the large geographical differences of the Eastern Regional office, we have established an Eastern Regional Systems Manager, who operates under the direction of the Corporate Systems Manager. The Eastern Regional Systems Manager provides support to the computer systems located in Versailles and Aberdeen.

The HP125

The remote locations throughout the world, which are referred to as satellite facilities, also had requirements for data processing on a local basis and the need to communicate to their individual regional office. To meet this requirement we looked at a variety of options and systems, and chose the HP125, which can operate as a stand-alone micro processor and also communicate to the HP3000, as the best choice to meet our needs.

Most of the Teleco applications, both financial and operational, require the collection and processing of data at the remote locations, transmitting that information to the regional office for additional processing and regional consolidation, and then transmitting that regional information on to the Corporate Office in Meriden, CT for final processing and reporting. The HP125, with its stand-alone ability to provide word-processing and other business functions such as Visicalc, and the Link 125 utility allowing it to operate as a remote communication device, was the perfect fit for our requirements. During data gathering at the remote locations, the 125 operates in a stand-alone mode, servicing both operational and financial applications. When it is not being used in a data entry and editing mode, it is used by the secretarial staff as a word-processing terminal. In addition, the financial staff are able to use Visicalc as a financial tool.

The hardware configuration at the satellite facilities consists of an HP125 with 64K Memory, an HP9135A Winchester Disk Drive, containing 4.4 million bytes storage with 248K floppy disk storage, a 2601A Daisywheel Letter Quality Printer, and a 3451P Racal Vadic multimodem. The software available on these systems is Word/125, Visicalc, Link/125 and Cobol II.

Determining an application development strategy involved evaluation of established skills in the CIS Department and capabilities of the various products on the CP/M market. Since the primart use would be business applications, we chose Microfocus Cobol II. This decision along with our staffs' existing knowledge of Cobol provided the elements for successful project development and implementation.

Following certain conventions our programmers were able to code and test Cobol programs on the HP3000 and then pass the source code to the HP125 for compilation and final testing. The Microfocus Forms II product was also used and proved to be extremely helpful in generating screens and skeleton programs for custom applications.

Applications such as Accounts Payable, Field Reporting, Inventory Control, Quality Assurance, Budgeting, General Ledger, and Invoicing were identified as requirements for the HP125 and are currently being evaluated for purchase or internal development.

The following satellite locations were scheduled for HP125 configurations:

Ventura, California

Anchorage, Alaska

Gabon, West Africa

Stavenger, Norway

Singapore, Republic of Singapore

Cairo, Egypt

Abu Dhabi, United Arab Emirates

The Development Approach

Corporate Information Systems has as a major objective to meet Teleco's worldwide information requirements with accurate, timely and cost-effective systems. We felt the best way to meet this objective was through a combination of purchased application software and internally developed software. We did not want to re-invent the wheel and were committed to looking very carefully at commercially available packages to meet many of our business applications. Many of these applications could be purchased and used with minor modifications. Some others required more extensive modifications and design changes, however, we felt for some applications this procedure still offerred significant time-savings over total in-house developement.

Application packages such as General Ledger, Accounts Payable, Payroll, Fixed Asset Accounting, and Personnel, were purchased systems. All other applications were developed by our own staff.

The Conversion Effort

The only major application that we had running on our IBM System 34 was a purchased package for our manufacturing application. This package was written in RPG and had been running on the IBM System 34 for approximately a year and a half. We were faced with the problem of either converting this system to the HP3000, or looking at buying a brand new package. We undertook a study to look at the two different options, and after a detailed analysis, the purchased packaged approach was rejected as too costly, highly time-consuming, and required major adjustments for the user.

The conversion approach was selected as the most viable option. This approach was aided by a conversion utility package, which we purchased from an independent software house, McDonnell Enterprises, Inc., that provided assistance in converting IBM RPG to HP3000 RPG. This utility proved extremely effective, and significantly reduced our conversion time over what we had originally estimated. There were approximately 378 programs converted from the IBM 34 to the HP3000 in approximately four months, utilizing two programmers. Because the conversion required very little modification, the impact on the user organization was non-existant.

The System Software

We are committed to central control of all software development and testing, and the software systems that are available at the central site on the Series 64 consists of the following products: Image Database, Cobol, Fortran, RPG, Dictionary, Decision Support Graphics, Inform, HPWord, EasyDraw, EasyChart, DSN, and MTS. Additional software products purchased from third party vendors include Spread, which is a financial modeling and budgeting package, Opticalc, which is an electronic spread sheet utility, and Textpro, which is a word processing software product that can be used from any HP terminal.

The User Information Center

At the Corporate office in Meriden, CT, we have established a User Information Center whose purpose is to provide data processing access to the users throughout the corporation who do not have access to a terminal at their location, or the terminals which they would normally use are unavailable. The information center contains a 2623A graphics terminal, with an HP7221C Plotter, three 2624B terminals, and one 2601A Letter Quality Printer. The software support that is available at the information center includes EasyDraw, EasyChart, and Decision Support Graphics for use on the Graphics terminal, word processing capability through Textpro on any of the terminals utilizing the letter quality printer as output, and access to any of the application products through the four terminals. All utilities and products such as Spread and Opticalc, TDP, Inform, etc., are also available. In addition, the HP125, which is the back-up and development system for the 125s in the field locations, is also located in the User Information Center and is available to users on a lower priority basis.

The Word Processing Environment

Teleco had a significant need for word processing capability, and at the time of the formation of Corporate Information Systems there already were some stand-alone word processing systems being used throughout the Corporate Office. We immediately identified the need to control the proliferation of word processing systems and insure compatibility and standardization throughout the company. We evaluated the several alternatives for meeting our word processing needs, which included stand-alone word processing stations offering a variety of capabilities, linked word processing stations, and word processing supported through the central computer.

After a detailed survey of our word processing needs and a careful analysis of the cost benefits and capabilities offered by each option, it was determined that the word processing approach best suited for the long-term growth at Teleco would be to support the word processing application through the HP3000.

This decision was not a popular one with all potential users since many of the users had seen some of the stand alone systems, such as Lanier or Wang, and felt that they offered significant capability above what was currently available from HP through HPWord. After a detailed analysis, we determined that HP was committed to office automation products, particularly HPWord and that many of the shortcomings currently in the product would be corrected as well as many of the special features found in other systems would be available in the near future. Compared to the stand-alone units, the cost benefits of choosing the HPWord option were significant.

We also determined that we had need for two levels of word processing capability. One, a more sophisticated need requiring the special features that were announced in HPWord II, and a less sophisticated requirement for general word processing support. HPWord and the 2626W word processing terminal were seleced to meet our advanced word processing requirements. We evaluated HPSlate to serve as a less sophisticated word processing product, and rejected HPSlate mostly on the basis of its lack of wordwraparound. The word processing product evaluation group, consisting of a majority of clerical support personnel, strongly agreed that in order for a product to be considered as a viable word processing system it must have word wraparound.

In looking at other third party vendor products, we determined that a product called Textpro, from Pantechnic, was the best system to meet our needs. Textpro had significant features and advantages over HPSlate, and could be run from any HP terminal. This immediately gave us word processing capability at all of our 65 terminal locations throughout the Corporate Office. Textpro was made available to all the regions to support their word processing requirements.

In addition to providing word processing capability for all secretarial locations, we also established a Word Processing Center with a full-time dedicated word processing operator. This center provides secretarial support for all departments throughout the Corporate Office. The center uses the HP 2626W terminal and HPWord. We have found the utilization of the word processing center to be very successful, and it adequately meets the typing work overflow from the various departments. It has also reduced the need for additional clerical staff. As the utilization in the word processing center increases, additional word processing operators will be added.

The Evaluation

To date, we have been extremely pleased with the performance of the HP hardware and software. The one area where we do feel that HP did not perform as promised was the word processing product, HPWord. In fall of 1982 HP told us that HPWord II would be available. At this time HPWord II is still not available to us and we have been under considerable criticism from the secretarial staff having promised them the capabilities in HPWord II for approximately nine months. The current problem with Word II to our knowledge is not the product itself, but that it requires the Qmit Version of MPE IV which currently is not available.

We strongly feel that compatibility among systems is key to successful and cost effective growth. Where HP products have been utilized they have performed successfully and have adequately met our needs, yet we felt it was important where the HP product did not meet our requirements, that we look to other products available from third party vendors.

To date, the policy of centralized control of all data processing has been very successful, and we feel that we are meeting Teleco's worldwide information requirements with accurate, timely and cost-effective systems.

Conclusion

Over the next few years Teleco is anticipating significant growth, both in its worldwide operations and in its computer configurations. We feel our selection of HP has been a proper one and that our computer systems will be able to grow as we grow. We are convinced that HP will continue to improve their product line and will be a strong computer partner for Teleco as we move into the future.

FOR THE MIS MANAGER:
EVEN IF LIFE GIVES YOU LEMONS,
YOU CAN MAKE LEMONADE!

Margaret F. Van Vliet

Vice President/Information Systems
Marsh & McLennan Group Associates
Western Region
San Francisco, California

ABSTRACT

Even if the MIS manager has more than his/her share of problems, it is possible to contribute to business productivity gains beyond improving programmer productivity and systems utilization. Given "lemons" the MIS manager can make "lemonade." The key is to create a picture of how things could be, to provide a context in which decisions are made along the way to getting there. Formal long-range planning is important, but there is a more playful approach to dealing with the inevitable blocks to progress and productivity.

With a background in Systems Development and Data Base Administration, three years experience as MIS Manager of an HP 3000 installation have provided numerous illustrations of effecting change in an environment where systems needed updating and consolidation, where users were resistant to change. Some of the discoveries were: Often the way to go fast is to go slow. Follow the path of least resistance to get the most out of your resources; resistant users may become more receptive in time. Match people to task. Hire "doers." Direct the "bulldozer" personality into action. Send the "Teddy Bear" to make peace with the users!

BACKGROUND

I will begin by giving you some background to set the scene for what I will share with you. A lot of the ideas represent my personal management philosophy and come from a variety of sources and experiences. I have used examples from my experience and hope that those about whom I talk will think that there is value in my relating these to you. It is in the spirit of "creating a better world."

Marsh & McLennan Companies is the largest insurance broker in the world. Marsh & McLennan Group Associates is the subsidiary which provides group insurance products and administration. I am Manager of Information Systems for the Western Region of MMGA. Our headquarters are in San Francisco where there are approximately 100 employees. Within the western and mid-western United States, we have a number of smaller offices which range in size from three to over a dozen employees. Sometimes I think that we have the best of both worlds - being a part of a large corporation and working directly with a small number of users.

At the Regional Data Center, we have an HP 3000 and a data processing staff consisting of seven people involved in programming and two in operations. Our major application systems provide billing and insurance administration support for professional association and trade association clients, accounting general ledger support and claims statistical reporting.

My particular mosaic of qualities and experience includes a degree in music theory and composition, eleven years as a housewife, then getting a degree in computer technology and entering the programming field - that was nine years ago. My first six years in data processing, I spent working in large-scale computer environments in systems analysis and data base administration. Along the way, I completed a master's degree in "Technology of Management." Three years ago, when I joined Marsh & McLennan, I had my first taste of management and working with Hewlett-Packard computers. As a complement to the professional side, I enjoy delving into psychology, "new age" management, metaphysics, transformation, and occasionally I become "Daisey the Clown."

IMPROVING BUSINESS PRODUCTIVITY

This conference focuses on ways to improve business productivity. We don't need to be reminded that better design techniques and productivity tools are important in improving programmer productivity. However, just as we can better effect system efficiency at the system design level than at the coding level, we need to be careful in deciding where to place our emphasis to raise business productivity levels. Two of my favorite phrases come to mind:

- There is no point in doing efficiently something which doesn't need to be done.
- If you don't know where you're going, you'll never get there.

With those ideas in mind, to get at what really effects productivity we need to ask questions like:

- Are we doing the right things?
- Are we utilizing our people to get the biggest return on their time and energy?
- Are we moving to support our company's business needs in the future no matter the current circumstances?

Computer Museum

Even if your company does not provide you with a written set of long term goals, you can create a picture of what you envision for your company and the systems which will support it. No matter where you are starting from, this provides the necessary "tension" for creativity - the chance to create reality in the future which is better than today. With people "walking in the same direction," stepping around the blocks in the way or using them to speed up the journey - that will increase productivity.

In my professional life, I have been faced with a lot of blocks, unpleasant situations, problems. Through it all, I have held in my mind the picture of smooth-running systems and harmonious, humane working relationships. In my personal life, there hangs on the wall an embroidered sign which says, "When Life Gives You Lemons, Make Lemonade." It was given to me along with a note which said: "To the best lemonade maker I know!" Obviously, that was the inspiration for my title, and it reflects part of my personal management philosophy.

ENVISION THE WAY YOU WANT IT TO BE

It is important to have a picture or vision of how you want things to be "down the road." I have found that it doesn't have to be a very detailed picture. As a matter of fact, knowing the essence without the specifics leaves room for flexibility. It is important to see the whole picture working together. There is no point in seeing an elaborate system without people working with it harmoniously! I am not pretending to provide specific techniques for reaching specific goals. (There are books written on the subject.) What I do want to illustrate is the usefulness of even a simple picture of a more ideal future in making decisions along the way to getting there.

Here are some examples of the kind of pictures I have had in mind while at Marsh & McLennan and some of the results indicating that we are moving in the right direction.

o Envision: Systems consolidated into an integrated set of systems with like functions consolidated and interrelationships defined. (The picture is true even if all parts are not automated at a given point.)

In three years, we have moved a set of old subsystems from a service bureau to an existing in-house system, installed a general ledger package, developed a generalized system to replace some client-specific systems, converted a data collection system from non-compatible hardware to an HP125.

o Envision: A "communication network" connecting people throughout the region to provide a greater degree of interaction. (This has begun without the hardware "how" being known.)

We placed HP125's in four offices in the Region. This gave us guaranteed compatability and encourages interchange of ideas. HP125 applications have been developed using Condor and have been transported to another office. One outside office is now an on-line user of one of our HP3000 application systems.

o Envision: A small professional Information Systems staff which analyzes business needs and participates in determining and implementing the best solution to support the business. (This might include enhancing existing software, selecting packaged software, improving procedures.)

The quality of personnel in the department has grown. The perspective of the staff has gone from seeing programming as the assumed solution to actually demanding a report writer.

On a smaller scale, a good example of the power of picturing a situation one way and then another comes from my data base administration days. One of my fellow data base architects tended to be pessimistic and always prepared for battle when going to a meeting with programmers to work on data base design. Through a combination of demonstration and words I convinced him that preparing for battle practically assured that there would be one, and that if he saw the meetings as being productive, cooperative ventures, they would be more so. After a couple of successful meetings, he said to me, "The analyst and I come out of meetings smiling these days. Our boss doesn't believe it!"

TAKE WHAT YOU GET

Whether or not you have a clear picture of where you are going, life will present you with opportunities and problems. At that point, all you can choose is your reaction. The challenge is there: to let the situation sour or make the best of it.

When I first joined Marsh & McLennan three years ago, they were just weaning themselves from the consultant who had provided most of the data processing support for some time. There were the beginnings of an in-house data processing department with two operators and four on the programming staff. I was the "new kid on the block" in the office, the only woman manager, a "foreigner" from the East coast and considered part of the enemy (being a hiree of the new head of the Western Region.) Two of my employees were men with management experience, of which I had none, and seemed out to "do each other in."

In the beginning, I saw changes which I thought would help streamline the operation, and I needed to feel I was "earning my keep." Often as I set out to give users the benefit of my wisdom, the fog of confusion would fill the room or the idea was rejected outright. My favorite story is that I suggested they use a common billing form instead of a different one for each client. It seemed perfectly logical to me. However, it was met with unexplained resistance. So, I gave up and backed off. A year and a half later, one of the users suggested it, and it was implemented. My discovery was: *When you feel resistance, back off; pushing may create more resistance.* A corollary is: *Often the way to go fast is to go slow.*

WHAT BEFORE HOW

Not only do you have to "take what you get," but consider the importance of the cardinal rule: *Determine what before how.* Consider how important it is in defining system requirements, designing data bases, in problem solving. It supports the idea of having goals or a clear vision before deciding upon a course of action.

One example of what vs. how comes from a meeting two of my programmers had with users who were responsible for a growing department and often fell into a "crisis" mode of operation. One challenge was to keep from merely reacting, putting ourselves into a "crisis mode" as well. Another was to keep the communication channels open, to keep resistance from growing. This particular meeting was to solve a problem with the system's data base. As the programmers presented a detailed list of *how* the problem needed to be solved, the users seemed uneasy about being told what to do. After the meeting, the programmers and I talked about the problem and decided that maintaining the integrity of the data base was *what* was most important. With a little thought, they came up with another way that they could have achieved that. Had they offered the users a choice of *how* we did *what* we needed to do, we might have settled on a mutually acceptable way, leaving the users feeling more "in control" and more cooperative.

An important point about first clearly defining the "what" is that you then leave room for brainstorming all kinds of possible solutions, finding one which will work and meet everyone's needs. Then you get to try the "how" you selected. If it doesn't work, you get to try another. That's part of creativity. Remember: *If something isn't working, try something else.*

In the data processing field, we are always fighting the battle of producing adequate system and program documentation. What happens when we try the "what-how" test? Clearly, it is important to determine what we are trying to accomplish by the documentation. At some installations, the answer must be to fill volumes. To me, documentation should provide:

- What is necessary to force thinking and communicate information during the design of a system. This is working documentation and does not need to be "maintained."

- What is necessary for programmers to maintain the system. How that is done could be decided based on the easiest to produce, the clearest, the easiest to maintain (the closer to the "source" the better.)

If there is one idea I would choose to instill in my people, it would be this determination of what before how. I cannot tell you the number of times I have been reminded of its simple power to give the necessary perspective to solve a problem or approach a situation constructively. Keep it clearly in mind.

LISTEN TO YOUR PEOPLE

I am a "people person," a bias I am happy to admit. Although I am aware of the importance of structure, follow-through and other management skills and tools, I often provide people perspectives as my contribution to the whole organization working. I believe that it is possible for individuals to meet their own goals and to grow professionally while achieving the organizational vision. That is part of my vision. During an interview with a young woman who subsequently has become a part of our staff, I began telling her a little bit about each person in the department. As I was going down the list, she stopped me and said, "You really like your people!" Yes I do.

An example of a situation which served the person and provided increased productivity for the organization, involves one of my programmers who gave birth to her first child a year ago. After recovering from the birth, she began working at home using a terminal and modem which we provided. She did this for several months, coming into the office once or twice a week to pick up listings and discuss assignments with her project leader. Although we didn't know how it would work out, it certainly seemed worth a try. The result was that the young programmer blossomed, learned to work more independently and was highly productive.

One form of "listening" that I have found useful is just noticing when symptoms of "something is not right" surface. In one case, I assigned a project leader to another project, because he was becoming increasingly impatient with a conversion that had been drawn out and a user who was difficult to deal with. Although it may have slowed the completion of the conversion project, it did give a more relaxed junior member of the team a chance to complete the project and relieve some built up tension.

By noticing another programmer's decreasing enthusiasm, the project leader and I decided to give her responsibility for working with a user department as they began using a new system. She has arisen to the cause, and we now have to look for signs that she is putting too much time and effort into her job! Another programmer, who had worked on maintenance of one of our systems for a year and seemed ready for a change was assigned to another system, thus giving us greater depth of support and the programmer a chance to learn a new system.

We sell computer time to another subsidiary, whose contact person is a young woman who had been abrasive and persistant to the point of being obnoxious. Our lead computer operator, who joined our staff a year ago, has a wonderful way of dealing with people, so the complaints had lessened. At our recommendation, the young woman had attended an HP operations class, and when we needed someone to be in the computer room one afternoon, we invited her to "fill in." From what I can tell, it had a miraculous effect and she is now one of our most cooperative users. Based on that experience, in each user department we are identifying a data processing contact person who will be given additional training by our operations staff and will relay information to and from computer operations.

USE YOUR PEOPLE

Not only do you have to listen to your people, but carefully matching personalities with the task at hand can be very rewarding. When I ran into resistance with the user community, I pulled back from the front lines and more or less patiently took a backseat. (I should mention that the Director, for whom I work, has been extremely supportive and patient as I have carefully moved my way through what has sometimes felt like a mine field.) Fortunately, we were able to move a person with a "Teddy Bear" personality in to support the major billing system, around which there seemed to be the biggest blocks. He has done an incredible job of gaining the users' confidence and respect. The users of the system are now initiating requests for changes to their system.

Another member of our team is an absolute "bulldozer!" Although he has other qualities, I can be sure that if we need swift and sure action, he can be counted on to get the job done.

Not only is it important to match people to the task, but there may be no greater productivity drain than trying to develop systems without the support of the users involved. At the time when the users of the primary billing system seemed resistant to any change or suggestion, although it was perhaps the logical choice for redevelopment, it would have been frustrating and a waste of time to attempt to do more than maintain it. As a result, we chose to develop a system whose users were anxious to have system support. While we were working on that system, the users of the other system were becoming ready to accept changes to their system. A key idea in achieving increased productivity is: Use your resources where they will allow you to achieve the greatest productivity.

KEEP IT LIGHT AND SIMPLE

We are all in the cast of characters in this drama of life, which seems awfully serious sometimes. Keeping a sense of humor can be almost a necessity. Play a little. My programming staff spent considerable time picking out just the right name for their new system. They decided on "Eagle." I'm not sure why, but as an American, I can't complain. Since then, we have acquired a couple of stuffed toy eagles, two of which have been presented to users, and it is amazing how many advertisements, pictures and posters we've found to add to our collection.

Complexity surrounds us, and often we try to fight complexity with complexity. That's a difficult one to win. Simple methods of tracking project progress, simple methods of documenting systems, simple solutions to problems. I'm for them all. Taking the concepts from the complex large-scale data base environment and applying them to the small-scale environment has been a challenge.

To help prepare the users in San Francisco and some of the other offices for increased use of automation in the future, we purchased HP125's armed with word processing and Visicalc software. By making sure that the hardware and software are compatible we have made our job simpler from the near-term and long-term support perspective. Users help each other out. People from one office travel to another to receive training and make important connections. Applications developed in one place can be used in another.

When you deal with computers, you deal with change. Change sends ripples through an organization and strikes terror in the hearts of us all, at one time or the other. From a "light and simple" perspective, one of the most effective tools is "seed planting." Whenever I get the chance, almost without thinking about it, I will throw out ideas to anyone and everyone. If you can give up having to be right or to get credit for an idea, watch patiently, and you will find people owning at least some of the ideas and putting them into practice.

To support movement toward the vision, I have found it useful to interpret what we have done and what we have had to do in terms of the overall vision of what we are after. Why not show that you are moving in the direction that you intend to, to add to the momentum? In preparing a five year plan last year, it was important to emphasize the progression from an environment where order was lacking to one in which there is an increasing sense of order. Concepts like the "communication network" were introduced, although we do not have concrete plans of how that will be implemented. We presented the conversion of a "Brand X" micro-based data collection system to an HP125 using dBASE II as providing a "prototype" system and an introduction to systems for a new user.

SUMMARY

There will always be problems or situations that are less than ideal - that is life. If you approach your role as the MIS manager by creating a vision of how you want systems and people to interact in the future, you will find that you naturally make decisions in that context and thus insure that you move toward that goal. If you look at problems in the context of your vision, you may see them as lemons just waiting to be made into lemonade! It is even possible that you may enjoy the process and more importantly move toward that future you have envisioned.

FOR MORE INFORMATION AND IDEAS

These books and workshops offer a variety of information and ideas. They are some of my favorites.

The Acquarian Conspiracy: Personal and Social Transformation in the 1980's by Marilyn Ferguson. (J. P. Tarcher, Inc, Los Angeles, 1980.) This is a look at the transformative process which is going on in the world today.

The Courage to Create by Rollo May. (W. W. Norton & Company, Inc., New York, 1975.) Looking at the creative process shows that the discrepancy between an idea and current reality creates tension necessary for creativity.

Getting to Yes - Negotiating Agreement Without Giving In by Roger Fisher and William Ury. (Houghton Mifflin, USA, 1981.) It provides good ideas for separating people from the problem and allowing everyone to win.

Leadership and Mastery Workshop. (Innovation Associates, P. O. Box 2008, Framingham, Massachusetts 01710.) This three-day workshop presents concepts and techniques which support the use of intuition and reason in providing leadership for inspired performance in an organization.

The Little Prince by Antoine de Saint Exupery. (Harcourt Brace Jovanovich, Inc., New York, 1943.) If you do not know this story, read it. "There are few stories that in some way, in some degree, change the world forever for their readers. This is one."

More Time Workshop developed by John Gavin. (Information available through the EST Communication Network, San Francisco, California) A unique twist on time management which begins with the fact that you can't manage time, but there are all sorts of ways you can reclaim energy which has been drained.

Up the Organization by Robert Townsend. (Fawcett Crest Books, USA, 1970.) A witty and wise book which is full of ideas about management and organizations.

Systems Development, Projects Life - The practical experience.

by

Carl Christian Lassen, M.Sc.

Introducing computers to the administrative processes and systems for the planning, analysis and control, can be done in many ways. During the years the community has seen methods, models and developed tools to monitor the development proces. Almost any project has had crisis during the development phases, but very little has been told about these problems.

The paper comments on the threats to the succesful implementation in the generations of computer development.

The paper is segmented into "Comments on .."

- The E-system, the reality for future applications
- The central batch processing
- The first 'on line' query systems
- The first 'real time' systems
- The next distributed and multifunction systems
- The first 'hybrid' system using corporate networks, mikroes and the evolution of 'packages'.

Basis for the presentation is the development of the 'E-System' - a result of the fifth generation experience, which includes the development towards systems which are independent of the hardware to be used or operating system, file system, compilers and communications standards.

The E-system, the reality for future applications.

The skills developed during the experiences which are briefly commented in the subsequent paragraphs, helped us to design a new system.

The E-System is a package to be used by many different corporations because it includes all normal functions which are used in the business administration.

The experience with the use of data bases, index files, direct accessed files has also created the internal file structure and acces method which is far more effective than any other method.

The experiences with many different brands of terminals and terminal handling methods has set the standard for the future demands to be accepted.

The experience with the security provisions common to many operating systems and not satisfiing the needs for the combined funtionailty within a corporation has set the standard for the internal access control.

The experience with the many different computers and operating systems has helped to seperate machine or language dependent functions from the applications modules.

The succesful help of certain methods for backup and recovery from accidents and for tele communication problems is the basis for the design of the logging funtions. Included in this design is the Audit Trail, which insure that data can be retreived and audited.

The E-System is the first hybrid system giving the flexibilty which is the wish of many corporations. Based upon the experience and the skills - the goals for the development was set in the spring of 1983, the ideas and the realisation is in front.

Comments on the central batch processing.

The central batch processing system was designed around the master files to be updated. Often you had a few tape drives and with a system having one cpu, four tape drives and perhaps a small disc drive, a printer and card or papertape reader, you mastered the administrative processes from mid 60's. One transaction file as input containing many record types. One master file as input, and an updated masterfile as output. Also output for updating other master files was created or copied from the input transaction file to the output. andwere written to an output transaction file, which were input to another process. The master file was only accessed in one program, and therefore many rules were given to perform the updating in a logically correct sequence.

The major threat is the problems encountered because the users had to remember many odd rules regarding processing sequences in order to understand the figures listed on periodical lists. For example you had to decide if incoming goods had to be dealt with before the sales transactions or if the opposite sequence should be followed. This had nothing to do with the physical handling of the goods, so in many cases figures on lists did not show the true picture.

The first 'on line' systems.

In order to overcome some of the misunderstandings the weekly lists were replaced by terminals to be used as inquiry tools against files with data as new as the latest update run.

The first attempts to read more up to date information was taken by the bankers, They first looked into the master files and then in the transactions file, which would be used during next update run. The Key-to-Disc data entering methods gave the clue.

- The solution will satisfy many needs, but the data are not 'real time' correct.

Comments on the first 'real time' solutions.

The introduction of minicomputers in the beginning of the 70's and the declining purchase price for terminals gave some answers to the need for a real time processing. The balances could be updated immidiately and the paperwork for the physical handling could be printed at the same time.

Looking through the documentation you will find that the thinking about coding structures and methods followed the rules very similar to that of batchprocessing. The programme should contain all functions to be applied to a given file - both the heavily used ones and the seldom used. The complications became tremendous as the input filters had to look for many errors and the programmers felt that the users needed as much information as possible. Nobody asked how much of the information was relevant. The syndrom was - "You got information in the file - Show it !" Most of the programmes were coded as extended batch programmes, internal switches were used in order to try to control the flow through the procedures.

After the first sad experiences with programmes which were impossible to maintain because everything was coded inline, thus making the code unreadable, the profets talked about modules, structured code etc.

In this generation the major threat remained to be the misunderstandings as regard to the correctnes of data. Users felt that data were private to them and that updating performed by others was difficult to accept.

Many hardware suppliers overestimated the capabilities of their hardware and software. Unanticipated expansions of memory were common experience.

Comments on the second generation 'real time' solutions.

Following the evil discussions with users about the correctness of data and the difficulties in reading the programmes and the frustrations given by the fact that there exists no insured connection between the programme and the documentation due to the lack of time to finish this within budget, a few solutions could be chosen. The Structured - 'GO TO' less programming philosofy was one. The absolute seperation of the logical functions into modules or programmes was another one. In this case you would use one module (Programme or procedure) to perform a given task. Then it is easy to be sure where data are changed in a well defined way. And it gave the user the strict knowledge of where and when data were updated. It also gave him the control of who did the updating.

One task - one programme.

By coding a programme to perform one and only one task, you introduce a range of programmes. It is easy to communicate with the users about a particular function, and you can distribute the coding among many persons, as there wil be very little need for internal coordination. But there is introduced a considerable risk for the total installation for degrading the performance as seen by the users. In the analysis phase of the project, you are told about the anticipated use of the functions, ie how many times a day they are expected to be used. If you know it is true data, then you know the risk and can prevent any damage, but if this is not the case, your risk is that they use the functions many more times. Swapping from one function to another could force the closing of files, databases, even cleaning some workspaces etc before the new programme will open the same files and data bases and create clean workspaces. This is a costly process for a computer to perform, extensive access to the file directory, looking for free space on the discs, controlling the acces modes against other users or the passwords supplied against the security provisions. If you change functions many times a day the cost of the extra coding to include more logical functions within one physical module will pay. If only a few times, it pays to code the one only physical function, and if you use it for training of new programmers, there will be great benefits because the risk for error is high, but the cost to find it is very low. The experiences shows that the users underestimates this swapping reportedly at a factor of 2.5.

Introduce process handling to the 'Maestro' programme.

In the DOMI Spare Parts Project a solution was to introduce process handling within the most used programme. The users ran this programme after the logon. This main programme creates a son process when the user wish to use another function. Finishing this special task, then the son process terminates and the main programme takes over. This very simple change cut the openings and closes for files etc to one third and - naturally - had a tremendous effect upon responsetime and elapse time on batch processes.

Comments on the distributed and multifunction systems.

If your application is designed to support local users and users who are working within few of the range of applications, ex. order processing or accounts, then the first generation 'real time' system will satisfy many needs.

The introduction of terminals to be connected via tele processing equipment, modems, multiplexers, data exchanges and the like, or the fact that the users will have to work within many applications at the same time creates the important need for rethinking the design of the applications.

Using public networks, telephone or datanet, restricts the speed of transmission between the terminal and the computer. It becomes prohibitive to 'paint' full screens more than very few times during the day, and the screen has to be used for a long period. It is impossible to change between a set of screens because the transmission of these at 1200 - 2400 baud wll take seconds after which the user enters data in a few blanks, transmit this and wait for the validaton and the answer to refill the screen, then again enter the acknowledgement, blank the blanks and enter new data. In many cases the user wish to change the screen to another and a long wait is introduced. Solutions to overcome this could be to merge many screens to one common layout or to buy more expensive terminals to store the screens locally and just paint on demand from the application. In both cases the task of coding the programmes becomes more tricky and risky.

Applications designed for departments with few tasks could use first generation 'real time' methods. When the size of the company to support declines, there will be very few persons to do the data handling. DOMI supports large departments and - within the same system - also outlets with only one or two persons performing everything. This means that they will have to use the whole spectrum of functions. They will prefer to use them true 'real time' with no waiting time, and they will switch between the functions for every new transaction. This forces the application to be capable of dealing with every file in the whole system and to support every transaction type within one programme. The D-system at DOMI contains ca 120 different 'real time' functions, the user accesses 15 databases and files, giving a total of 75-85 physical files. The average use of a function before another one is used is as little as 1.2 entered transactions.

The method is to use of one Main Programme which calls the demanded procedures from the Segmented Libray - SL. The main programme reads the terminal and decides which function the user actually uses and call the relevant procedure. The use of a user controlled data entering method - Key the transaction as one line of characters using a field delimiter sign, or ask the programme to prompt by field for a defined transaction type - gives the fleksibility to cope with the different experiences amongst the users. In paranthesis - the main programme also supports V/Plus screen handling, but we did not install HP terminals or wanted to create the overhead involved. This system supports any terminal which can be hung to the HP3000.

This generation uses methods for datacommunication build around public telephone lines, modems, multiplexers etc. It saves a lot of money to use these boxes, but the quality of transmission is only fair. As very few organisations employ persons with considerable knowledge about data communication the reliability is a must. And it is not the experience with these systems.

Comments on the first generation 'Hybrid Solution'.

The 'Hybrid' solution includes the elements of a central data center, remote minies and mikroes, a corporate network to connect these locations.

The 'Hybrid' solution also introduce the flexibility to include hardware and software from many different suppliers.

The 'Hybrid' solution gives the user the flexibility to decide if access to central files should be employed immediately or could be deferred to next time more important data are to be transmitted.

New technologies -Packet Switching and Circuit Switching networks will replace the leased or dialed up lines and offer better quality, higher speeds and sevices. The design of the applications will change.

The outstanding develoment of microprocessor technology giving local performance and storing capabilities at fairly low cost will force the applications designer to rethink the methods to be used when data are to be distributed.

The facts about the need for data in the local environment compared with the corporate need for total information, are the criterias for design of the application.

The prospects of the use of the hybrid solution includes the capability to store data on a central system giving the corporate management the data for control and analysis and budgetting. The decentral user will have acces to his data through a network. Those users who always need instant response to a query could use a leased line, or a call through a Circuit Switching network. The local processor should be capable to decide when this communication could best take place.

The local processing could include the printing from a file to the local printer, entry of data which normally are controlled using hash totals, entry of data when validation could be deferred to a point where the call through the network is cost effective, the typical mikrocomputer tasks and logging of all activity on a local basis.

The result is that in the future, smaller outlets will be included in the network as the costs are very small. It will give the possibility to decide for every outlet how to perform the administrative processes based upon local needs and not on corporate system development strategies.

As compared to earlier system development methods, the complication of the new hybrid systems may be much bigger.

As result of these experiences we present the above mentioned E-System, the fifth generation total and high performing administrative system.

References which could be read in earlier conference papers or articles

"The way to the Right Solution"
Article by Carl Christian Lassen, Supergroup Newsletter July 1982

"Experiences with the SL Techniques, easing the Development and Maintenance Burden"
Paper by Hanne Hansen at the IUG Copenhagen Conference 1982

"How to get a high-performance Order-File"
Paper by Hanne Hansen at the IUG Edinburgh Conference 1983

"Centralized Data Base Access"
Paper by Eric Wendelboe at the IUG Copenhagen Conference 1982

"Solve your Problems with Imagination"
Paper by Eric Wendelboe at the IUG Edinburgh Conference 1983

"Neophyte in a Turbulent Environment"
Paper by Jens Ørum at the IUG Copenhagen Conference 1982

COMPUTER MEGATRENDS OF THE 80'S
AFFECTING BUSINESS PRODUCTIVITY

BY: Barry Klaas

INTRODUCTION

Rapidly evolving trends are painting the picture of what lies ahead in business computers. The megatrends in the computer business in the next ten years promise to jolt the thinking of many traditionalists. To those who are flexible, follow the trend and respond, business productivity improvements will result. To ignore these trends could mean failure. This presentation conceptualizes these major (mega) trends in the computer industry as seen from within Hewlett-Packard. An exploration of computer technology and its economics, coupled with shifting user requirements will remove some of the clouds as to where we are destined in the 80's. Those who are thinking and planning ahead in the 2 to 10 year range should find the information on the trends and conclusions drawn most interesting.

Megatrends...let's have a definition. Megatrends are the trends that are reshaping the world. Today we will cover those that have impact on contribution of the computer industry to improving productivity. Here is a list of the megatrends expressed in this session:

- The value of information is increasing and we are moving from an industrial based society to an information based world.

- The economic impact of new affordable technology coupled with an increased emphasis on individual contributions and self reliance, and democratic participation has facilitated a massive decentralization, a movement away from hierarchies to decentralized networks.

o The advent of "high tech" has led to a concentration of focus on the human touch. The adeptation of "high tech" is coupled to willingness of people to accept it.

o The resources of business are becoming more mobile and sharable on a global level. Data communications is a key to supporting this trend.

As supporting evidence that these megatrends are taking place, consider these facts recently developed by International Data Corporation. Let us examine the mindboggling state to which the computer has brought us:

o Over 5 million computers inhabit the earth. By 1987 there will be over five times that many. Computers are right now being shipped at the rate of one every 8 seconds.

o The collective electronic brainpower (instructions per second) capacity of computers installed doubles every two years.

o Five years from now there will be over 50 million electronic keyboard devices - computers, terminals, office systems... in the U.S., almost 100 million worldwide.

o Yearly worldwide investment in computer hardware, software, programming, and services is over $180 billion. By 1987 it will be close to $500 billion.

The advancement of computers and "high tech" is so rapid that as one wag puts it, "the wind blows in over California, picks up some sand (Silicon), a few hours later it is raining computers over the rest of the country."

MATCHING NEEDS TO CAPABILITIES

The evolution of computers has led to a wide range of choices for us. Some users prefer personal computers, some departments want to manage their own system, and division or corporate level systems require a broad range of power and functionality. Can these diverse requirements be tied together?

The successful computer vendor must match the organizational structure of our institutions. This means a matching of needs at the individual, department or office level, and at the division or corporate level and tieing these capabilities with common software, sharable data, and a flexible network. We are going to look at a diagram of this represented by three overlapping ovals on a price-performance matrix.

The first area represents personal computer needs. Prices for business P.C.'s typically start at $2K and fully configured P.C. systems reach up to $10K. The multiple user systems are above $10K today. The office oriented departmental systems support for 2 to 100 or more users and range upwards to $100K. The systems capable of supporting divisional or corporate requirements from HP are targted not to exceed $200K at entry level configuration and would support connection of up to 400 or so terminals.

Our analysis shows the prices associated with these areas to remain constant over the immediate future while capability and functionality double every 18 to 24 months. The more the system is associated with satisfying personal needs the more it must be perceived as a tool that extends and expands the individuals ability to do their job. Concentration on satisfying self-reliance needs, elimination of computer technical knowledge requirements, humanizing high-tech, and matching the learning patterns of the user will be keys to success.

The trend is to supply software across the organization for all these systems which is compatible at the application level. The benefit to the user is that he can transfer and transport what he has learned across these organizational boundaries. This, however, poses significant challenge to vendors. For vendors who create, supply and support the hardware, operating environment, and applications at each level, the task of transportability with compatibility is complex but easier than for vendors who concentrate on or control less than the whole spectrum.

MEGATREND #1

The 80's and 90's represent a "megashift" from an industrial to an information base society.

This is not a forecast but an extension of current fact. Less than 20% of the U.S. work force is employed in direct hands-on manufacturing. On a functional basis 60% of the same workforce produces or processes information as a key function of their job. Manufacturing has become mobile... able to shift from one geographic location to another in response to competitive pressure and facilitated by advanced replicated information systems. Traditional industrial production is falling off in the prime industrial nations but a strong surge of "high-tech" industries providing better solutions in making information available are filling the voids created by pre-computer age industry.

Information is replacing hard goods. The value of information is both supplanting and supplementing the value of goods. Just as information regarding nutritional value influence the selection of foods to meet more informed needs and enhances the value of the food, so too information about the changing needs of the customer influence the selection of what to produce and enhances the chance of a successful sale. A happier customer whose needs are being better met because of better information about the customer is likely to willingly pay a premium for that satisfaction.

So, cheaper manufacturing is moving the production of goods to new places in the world but the success of a product depends more and more on the information that tells whether customer needs are met and satisfied (as a recognition of what is valued and needed by customers and what satisfaction your product provides in meeting the values and needs).

MEGATREND #2

Every high technology introduction requires a clear identification of the human benefit or the technology is likely to be rejected or limited.

The advent of computer technology in the office and on a personal level can easily be met with resistance or rejection. Acceptance by large portions of the population of new technology requires developing a positive mental attitude about it and a willingness to adapt to it. Technology that creates more frustration or intimidation than satisfaction will be slow to advance.

Completing a job sooner, more complete communication, a deeper understanding of the basis for a decision, or more time for person-to-person interaction can be reasons for acceptance of technology. All highly successful computer products will be highly human engineered in the 80's. As a telephone (or a pen or a pencil) is considered a basic tool for office workers, so too should computer products be designed to be seen as easily acceptable tools to help get the job done. Computer based tools will be highly accepted when they are devoid of computer knowledge requirements and easily applied to a variety of tasks where the user perceives the high tech tool as an extension of their work environment.

MEGATREND #3

Centralized structures are declining. We are decentralizing and growing stronger from the bottom up.

Resurgence of productivity through computers is taking place at the level of the individual....whether that person is an engineer, a manager, a secretary, a programmer, or a president. Information is no longer clustered at one central source. With delegation of responsibility and accountability comes the delegation and distribution of information associated with it. In cascading sequence, information centers develop at corporate, division, department, and individual levels.

Top down solutions no longer prevail. The power and resources for dealing with problems are building at the grass roots level along with the computer resources available to the individual. Concurrent with the availability of more powerful computer based tools is a resurgence of self-reliance. Taking responsibility for one's own destiny in using information tools continues on the upswing.

Related to this trend is the fact that in the decade from 1977 to 1987, the acquisition of computers by departments other than MIS will have grown from 10% to about 40% of total purchases (based on dollar value).

MEGATREND #4

We are moving from hierarchies to networking. The computer evolution is leading to new management philosophy and information management structures.

Networking is a consequence of rise of the value of information and the increase in jobs which have an increased information content. Communications technology has kept pace with increased demand for more flexibility and functionality. Networking allows its participants to treat each other as peers because information has nothing to do with status in a hierarchy. Sharing of information between different levels and across organizations levels can easily shift an artificially contrieved structure to a more realistic cooperative structure.

INDUSTRY TRENDS THROUGH THE YEAR 2000

Now let's talk about in a global sense what's happening to the total industry and get the sense of the trends that are affecting us, you, and everyone else. We're going to look at a diagram that goes from the 1970's to the year 2000. On this axis we represent the dollar value of all installed computer systems, including personal computers to the largest mainframes.

In the 1970's, there were really only two segments to the marketplace: the mainframe marketplace and the mincomputer marketplace. What economists tell us is that through the future, the mainframe marketplace is expected to flatten out in terms of the installed value of the computers while the rest of the market will be expanding tremendously. Let's characterize this upperhalf. In the late 70's, the personal computer market began to grow. We expect that we are going to see tremendous growth for the personal computer market. Also, in the seventies, there was a substantial growth of superminis. These were really systems of mainframe power running in a minicomputer environment:

a less complex environment, a more friendly environment, easier to use and really akin to distributed processing concepts. As we grow in the next 20 years the supermini area is going to expand. As well, of the micro and mini areas are going to expand. In the minicomputer market, there are three of segments: the lowcost minis with anywhere from 2 to 40 terminals, the midrange with 40 to 100 terminals, and the upper end with anywhere from 100 to 200 or more terminals in the supermini area.

But that isn't the importance of the characterization that I want to get across. What's really important about these two areas is the dividing line between the mainframe and the rest of this marketplace. The mainframe marketplace requires substantial technical resources to make it succeed. That is, a lot of well-trained system and application programmers, and data communication experts will be required. These are people that are becoming more and more scarce. If you look at our ability to produce engineers, computer science people, and programmers in the future, we are really limited compared to the demands that will be made for these individuals. Therefore, in this marketplace which demands a great deal of programmer resources, the key to success is programmer productivity because you won't be able to get the number of these people resources you want..they will be scarce, they will be limited. Computer futurist James Martin has predicted that by 1990 computers will be executing 93 times as much code as exists today, resulting in a need for 27.9 mil programmers to create that code if there are no improvements in programmer productivity. Therefore, there is a major emphasis on productivity. It must be increased at least 2 1/2 times per year from here on out. Not only are programmers going to have to be more productive, but the software that is supplied to the industry has to become more productive, more readily available, more readily installable, and much more universally applicable. This applies even more so to the smaller less expensive systems in the market where instead of programmer productivity, user productivity is essential. User software, as well as systems, must be user-oriented and friendly with an abundance of "easy to use" features available.

Let's look for a moment now at the economics of what is represented in the marketplace. I'm going to contrast, if you will, a 1 MIP CPU with the cost of one person. The data used is an average from Hewlett-Packard including benefits. We're using the 1 MIP CPU here because 5 years ago when we were thinking about the design of the Series 64, we targeted it for its on-line power to be equivalent to the IBM 370/158. The 158 in the early 70's was one of the most powerful systems available. I'll position it in this diagram in the mainframe segment. By the year 1980, that system had migrated into the supermini segment. By the year 1990, the power of a 158 will be located in the low cost minicomputer segment. And before the year 2000, the power of a 158 will be located in the personal computer segment. There are a number of lines which go through these curves which represent migration of computer power as more powerful technologies and lower costs become a reality. Mainframe power will be migrating to a personal computer. In fact, if we use the analogy if the auto industry did what the computer industry has done in terms of price/performance, you'd be able to buy a Rolls Royce for about $100 and get about 20,000 miles to the gallon.

The cost of a 1 MIP CPU in 1970 was over $1 million. The cost of that same CPU in 1980 was around $480,000. In 1982, HP came out with its first 1 MIP CPU, the Series 64, at a price of about $220,000. By the year 1990, the industry forecast is that a 1 MIP CPU will cost $25,000. And by the year 2000, it will be priced under $10,000. Now, let's take a look at people. In 1970, the average cost for one individual was $16,000. In 1980, the cost had risen to $27,000. By the year 1990, we project the cost to be $49,000. And by the year 2000, we will be looking at cost being between $70,000 and $80,000 per individual. Think back. When you justified a computer back in the 1970's, what was the basis for your decision? The basis went something like this, we need a $1 million computer with the power to do the job. What we need in order to make this computer a very efficient and productive resource is a lot of these resources here, skilled people who can make this resource (computer) very productive. By 1990, there will have been a dramatic change. The value of the individual at a one year cost will be much higher than the purchase price of a 1 MIP computer system. Return on investment for computers will be based on improvements to the efficiency of people rather than improvement in computer efficiency.

The ability to get more productivity from people, your most costly asset, will be the focal point. The hardware of the computer in 1990 in itself will not, with it's low cost, provide the ability to leverage and get the most out of your people. What will provide productivity will be software. The whole industry will benefit from lower hardware cost. HP is following this path by lowering its cost of hardware at a rate of 25% per year, or providing at least 25% more performance at the same price. In fact, HP is one of the leaders in the price/performance race. By 1986 for example, we intend to introduce a 1 MIP CPU for $25,000 -- 4 years ahead of the industry forecast. We're involved with two R & D projects now in our labs which have produced VLSI circuits that have a half million devices on a quarter inch square piece of silicon.

This more than twice what our neighbors here in Silicon Valley are able to produce on the same chip. We are able to do this because we have made some breakthroughs in the design of circuits as well as in the equipment that produces those circuits. We have developed a new X-ray lithography device which is able to get circuit lines very close together, which is the limiting factor on how dense you can make a circuit on a silicon chip. We're able to narrow the circuit lines on the chips to be within about 2 microns (one millionth of a meter) of each other. With technologies like this, HP will be able to achieve these types of economics ahead of almost everyone else.

Not only is hardware going to differentiate leading vendors but the capability functionality, and quality of the software will have an even larger impact. Five years ago, HP invested about 70% of our R & D dollars into hardware and 30% into software. Today, it is exactly the inverse of that. We're investing 70% of our R & D dollars into software and 30% into hardware. Our contribution will be that the software will be exceptionally well integrated in the whole system.

These future trends tell us that those companies that will survive in this business are going to have to supply a lot of very good software that helps both users and programmers improve their productivity. These are areas where both users and suppliers need to focus.

Decentralising the DP function

Nicholas H. Cuthbert
Group Chairman
Jade Computing Group

Declining hardware costs, increasing self-confidence of users, and developing technologies have provided compelling reasons in the minds of many managers to bring distributed data processing into their organizations. Rather than let it spread almost randomly, managers who recognize the strengths as well as the pitfalls of distributed data processing are seeking to establish coherent policies and procedures. I shall define distributed data processing in terms of responsibilities for managing all phases of information processing. I will also provide a framework which companies can use to evaluate alternatives and to develop a distributed data processing strategy that matches their organizational structure.

In essence, distributed data processing involves the exciting prospect of assigning responsibility for data processing related activities so that both business and technical knowledge can be applied more readily to the achievement of an organization's goals.

The move to distributed data processing is possible today in large part because of the drop in cost of computer hardware. Mini- and micro-computers are becoming so inexpensive that it is quite reasonable for a company to buy three, four or even more small computers to do the work previously performed by one central computer or to automate activities previously performed manually. Whereas, in the past, economies of scale may have driven a company to use one large data processing centre to perform a wide variety of services, a company now might employ several different computers in different locations to provide more specialized services. Management can now arrange data processing facilities with much less concern for hardware costs.

Technical developments are also responsible for the recent practicality of distributed data processing. Major innovations, primarily in the areas of telecommunications and data base systems, have ushered in many new capabilities. Improvements in both hardware reliability and software have made possible the building of very sophisticated computer networks, and direct links from computer to computer are now common. Tasks which have been handled in large data centres may now be split into subtasks and these in turn may be farmed out to remote sites.

So to summarize the current position. Declining hardware costs, improving communications and data base capabilities, and the growing data processing experience of users are prime factors behind the move to decentralized data processing.

To take advantage of the power of decentralized data processing, managers need a clear conception of the complexities of using and managing the computer resource. Yet managers at all levels too often find themselves in difficulty because of their overly simplistic understanding of data processing, which they see chiefly as a collection of machines and technical issues; and oddly enough, DP managers sometimes are the worst culprits.

How can a manager avoid the pitfalls of the narrow view and systematically plan the placement of DP responsibilities? Primarily, he or she should recognize and work with two very fundamental assumptions:

(1) information systems should match a company's structure and strategy;

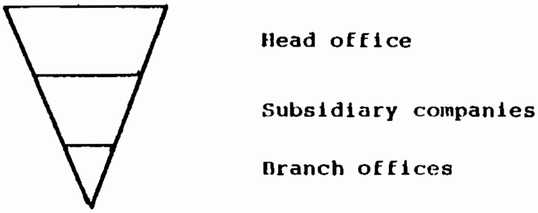
(2) activities that support information systems are wide ranging and seperable.

These basic notions are not only a good general way to understand the data processing resource, but they also allow us to build a framework for describing and planning distributed data processing systems.

To most technicians, distributed data processing simply means the spreading of computer hardware and data to multiple sites around an organization. This definition is deficient, however, because it overlooks a wide range of activities that help make information systems work, and furthermore it neglects the linkage of information systems to strategy and structure. In effect, its technical orientation leaves general managers out of discussions about the design of distributed data processing systems.

A broader definition acknowledges that data processing is an organizational resource consisting of many areas of activity, each of which may be executed or controlled by various individuals. These activities, or areas of responsibility, can be spread across an organization in a variety of ways, and a manager should carefully consider the appropriate degree of decentralization for each of them. Assignment of responsibilities by default seldom works.

Let us now look at the historical concept of decentralized data processing. As we are all aware computers were designed by scientists for their own use. The replacement of punched card tabulation equipment in companies by computers was an inevitable progression. Because, in the main, the punched card tabulation equipment was used as a replacement for accounting machines the computers came to be regarded as faster accounting machines. When the cost of computers started to fall they were used in subsidiary companies as faster accounting machines. Later still, with the advent of minicomputers, branch offices were computerized. The net result of this exercise was to distribute computer power as shown in diagram 1 below.



We must now ask ourselves, does the above structure reflect the structure of the company and its strategy. The answer is a resounding NO. It reflects the empire established by the accountants in the organization.

The relationship between structure and business plan has many dimensions. The key to a good design for an organization is not only the match between strategy and structure but the more complex match involving strategy, structure, and other administrative systems. Today many of these administrative systems are embodied in computerized information systems, which assist management in controlling and coordinating most of the activities of modern, large-scale organizations - tying together the efforts of diverse departments that pursue different but complementary ends.

Yet computer applications in general have been so specialized that many organizations - the data processing profession along with them - have overlooked potential, and more general, roles for information systems. Information systems are not simply labour-saving devices that support the activities of people in one or more departments. Rather, they are control and coordination devices that should fit an organization's formal structure and facilitate achievement of its business goals.

Most information systems are designed to play such roles, and it is worthwhile to re-examine them in that light. Consider the following typical situations:
- Production managers must maintain control over inventory levels to ensure that operations are not interrupted by sudden depletion of parts. They often establish this control by use of a computer stock control system, which, when coupled with formal procedures for requisitioning parts from the store, signals when inventory levels are low and in some cases even orders new parts automatically.
- Managers must coordinate the activities of sales with production so that when a large influx of orders come in production levels will rise to meet the demand. Managers often achieve such coordination by using an automated order entry system that processes sales orders and then, to fill these orders, updates the production schedule.

In the first case a computerized information system is used to CONTROL an important facet of operations, whereas in the second a system is used to COORDINATE different activities. In both cases the information systems fit the structural and strategic features of the organization.

To match information systems with the organization they serve, careful attention must go into planning the arrangement of the data processing resources that develop and operate information systems. The purpose of such a framework, in general, is to divide the overall problem into interrelated sub-problems so that each can be solved seperately while it contributes to an overall solution. A framework may also be used to provide a system of measures by which to compare organizations or assess the impact of organizational change.

My organizational model contains seventeen identified areas for which responsibilities must be assigned in varying degrees between the user and the central data processing group. They are:

- Hardware operation
- Telecommunications
- Systems programming
- Application systems maintenance
- Data base administration
- Application programming
- Systems analysis
- System documentation
- User training
- Providing security
- Setting priorities
- Standardizing tasks
- Accessing data
- Scheduling tasks
- Personnel planning
- Budgeting
- Evaluating products

Each of these areas must now be broken down into their items of responsibility and I have set out below three examples of this break-down into constituant items:

Hardware operation:
- prepare source documents
- manage data entry
- operate satellite processor without data base
- operate satellite processor with data base
- manage independent facility

Data base administration:
- control source documents
- determine data requirements and conventions
- carry out logical data base design
- carry out physical data base design
- set and enforce conventions for data access
- manage all data bases

Evaluating performance:
- collect performance data
- define performance requirements
- monitor performance
- determine appropriate response

In a centralized environment most of the items shown under each of the headings are administered by the DP manager and his staff. However as you decentralize these items progressively, from top to bottom, become the responsibility of the user.

To give an example of how the above framework is used, if the user has technical people who perform data base activities up to and including the physical design of the data base, then he or she is most likely responsible for logical data base design, the determination of data requirements and conventions, and the control of source documents. Similarly, the DP manager at some data centre performs the remaining tasks, setting and enforcing conventions for data access based in part on the design and, in general, managing all data bases.

One should recognize that management can control and co-ordinate activities not only by direct supervision but also by establishing comprehensive guidelines or standard operating procedures. The technical concerns of data base administrators, for example, are commonly controlled in this fashion. Their responsibilities can be either centralized in an individual or a group, decentralized but constrained by centrally designed standards, or decentralized with virtual independence. In the future, the second approach, that is, decentralizing the data base administration activity while using centrally established standards, will probably become increasingly important. In this way, the DP manager can define comprehensive standards that can be enforced by top managers or a highly placed steering committee and that can be used in a decentralized organization to protect the data processing department from excessive control by the user.

The framework just described equips us to look inside our own organizations to see to what extent each area of data processing activity is decentralized. By examining the complete set of activities, we see that decentralization of some tasks - be they technical or managerial - is a common occurence and that even classic types of information systems, such as stock control applications run on centralized computers, are supported in part by a variety of decentralized, or user, activities. Thus we can say that even these classic information systems represent a distribution of information processing responsibilities.

As we have seen, distributed data processing has implications that go beyond conventional approaches of applying computers to business. Although on the surface it appears to concern itself solely with the arrangement of computer hardware and data around an organization, at a deeper level it involves the management of a wide array of activities that go into the development and operation of computerized information systems.

Managers can organize these activities in many ways, and the choice they make should reflect the strategic and structural decisions they have made before. Thus, even if their information system design leaves hardware at some central site while assigning a few data processing tasks to users, their system is in some sense distributed.

According to this more general definition, nearly all computer applications are distributed. And, indeed, to ensure that information systems fit an organization, data processing should be distributed. Managers should concern themselves now with the question of how they can best distribute data processing for the information systems they require.

Information systems can support an organizational structure by strengthening communication lines and clarifying measures of performance. To achieve this, managers may choose to decentralize systematically the component activities of data processing within their organizations while paying close attention to overall business plans.

Once managers agree on the advantages of distributed data processing, where do they go from there? How do they plan for distributed computing to ensure that strategy, structure, and information systems match? Too often either management's planning sessions consist of intense technical advising, not to mention lobbying, or there is no planning at all. In some cases, management involvement and approval are carefully skirted. When managers are involved, often their thoughts are overly influenced by cost considerations. They tend to approve some plans solely on the basis of the potential for reducing operating expenses, and they tend to accept others because they are easy to use and do not require a staff of computer professionals.

Whereas promises of cost reduction or work simplification may be paramount in a go-ahead decision for a new information system, other factors are also critical, such as the availability of technical talent, the degree of coordination between users and programmers on project teams, and the careful construction and enforcement of standards. Unless managers have analytic techniques for assessing these other factors of a DP strategy, they will be severely hampered in their planning; success will depend on luck or on prodigious efforts to rescue a badly planned venture.

In developing systems to support administrative activities, companies tend to go through a process that reflects, among other things, management's growing sophistication in the use and control of computer technology and the organization's ability to use new management methods, a process often referred to as "organizational learning". The finding that the data processing budget of a number of companies follows an S-shaped curve over time prompted the hypothesis that there are four major stages of data processing growth, each with its own characteristics, problems, and opportunities.

The introduction of administrative innovations often produces these growth curves, and of course where we find them depends on what we measure and control and what kind of changes are possible. However, the use of distributed data processing as one form of administrative innovation has the potential of producing complex innovative disturbances, or dynamics, within the organization.

Let us now use my organizational model as a measuring stick to assess the way a company organizes its data processing resources. As we can see from the model, the data processing function involves many kinds of activities. Some are technical and some managerial, and the analytic separation of execution activities from control activities reflects this distinction. For execution activities, we then distinguish the technical tasks associated with systems development from those supporting systems operation. The result is a four-way split of data processing activities into execution of development, control of development, execution of operations, and control of operations.

It is probably easiest to illustrate the shift in control which occurs when an organization choses to move to distributed data processing by looking at one minute area within the data processing organization, that of hardware maintenance. With a centralized DP function the question of responsibility for hardware maintenance falls squarely on the shoulders of the DP manager. But what if hardware is procurred centrally for distribution to remote sites in several states, or in a European context in several countries. Who is then responsible for the maintenance of the hardware? I am not looking specifically at Hewlett-Packard now although two different problems do arise in the area of maintenance on HP equipment. If one purchases from a central point machines which are to be installed in say Brazil and Argentina then you are forced to decentralize responsibility for the maintenance of these machines. Similar problems could arise in Europe with the distribution of standardized software within an organization. For instance what language do you use on report headings?

Our organizational model can be used to clearly define the distribution of responsibilities in each area of DP operations. One has only to give careful consideration to each item listed under each main heading to arrive at a corporate policy for distributed data processing which sets clear and concise standards and controls for the entire organization. It is well worthwhile remembering that for each information system the policy may vary due to the degree of decentralization required by management.

How do managers decide how much to decentralize execution and control activities? Or when they do have an idea of the appropriate degree of user involvement in the design and operation of the information system, how do they achieve it? You will see from the examples given earlier that the range of involvement goes from essentially no involvement all the way to complete responsibility for a given system. We must therefore speak of user involvement as the combination, or sum, of decentralized execution and decentralized control. For instance, when users experience critical or constantly changing needs for specialized information in their normal business operations, a high degree of user involvement is probably warranted.

Let us now look at an example of the type of problem now facing most organizations. A marketing manager sees in mini- or even micro-computers a way to improve the performance of the company's sales representatives by providing them with more timely customer information. With this level of decentralization, he believes, local offices can maintain all types of data on their customers and thus create more responsive sales and service teams. Although the marketing manager has justifiable reasons for wanting greater sales force involvement in data processing, these concerns by themselves do not imply that local offices should have their own mini- or micro-computers. The requirement could just as easily be answered by a centralized customer enquiry system and remote terminals connected over telephone lines to that centralized system.

In conclusion, we have seen that distributed data processing offers great potential for improving the performance of individual departments and for promoting the success of an organization as a whole. Managers within these companies are turning to distributed data processing to help solve long-standing administrative problems and to help improve the control and coordination of fundamental business tasks. Enthusiasm is indeed strong for the decentralization of computing, and the call to acquire more computers is bound to increase.

Because of the demand, managers will be wise to plan carefully how their organizations will use information systems in the future. Along with these general plans, they should promulgate corporate policies establishing how they will arrange data processing resources to support their information systems. Only with such plans and policies can they ensure that information systems will fulfill their potentially powerful roles in the overall success of their business.

The author would like to thank, Alan Paley, formerly Director of Management Information Systems, Levi Strauss Northern Europe and Elizabeth Somogyi of Butler Cox Foundation for their suggestions in the writting of this paper.

Implementing a Manufacturing System in an Engineering Company

Stephen M. Foster B.A.(Econ), M.A.

1. Introduction

- the limited benefits from the use of a computer as an accounting tool.

2. The development process - a managerial view

2.1 design of the overall framework

- distributed processing
- selection criteria of the hardware/software
- the approval process
- creation and establishment of the 'right' systems team

2.2 the implementation process

- system design
- project control
- software quality assurance
- user education and training
- systems enhancements and maintenance

3. The advantages of a computer aided manufacturing system in an engineering environment

1. Introduction

In the 1960's and 70's most computer departments reported to the Finance function and solved accounting problems. It is a commonly held view and probably a correctly held view that although the introduction of computers more often than not reduced the staffing numbers considerably in the accounting and clerical areas, it required equally large increases in numbers of computer people: analysts, programmers, computer operators, card punch operators, etc., to achieve these savings. The computer equipment was also very expensive but at that time economic activity allowed both private industry and the public sector to be able to afford this expenditure, mostly to the benefit of computer manufacturers who are still wealthy today as a result. Private industry and the public sector is, however, not as wealthy today. But I will not attribute all of the blame for their decline to the computer manufacturers! Just some of it.

On a more positive note it is obvious that computers could and did perform tasks which would otherwise not get done. for instance their involvement in the space research projects.

At Alcan Plate Ltd. (APL), an engineering company engaged in the manufacture of high quality aluminium plate products, it did strike us, during 1978, that somewhere in between computers running batch payroll systems (old hat by now) and computers controlling, guiding and landing moon rockets (also "old hat" by now) there could be a computer solution which would assist us in running a more efficient business. Clearly since we did not have any computer equipment on site we should get the benefits of the computer technology that allowed men to land on the moon. We were aware that the astronauts didn't take any punch card equipment with them so although we knew very little about computers we knew we would not need any of these at Alcan Plate! But to what use would we put this new technology? We tried to focus on and define what we considered to be our major business problems. These were a lack of 1) good management information which resulted in the inability to control the business tightly and a lack of 2) good administration and production control systems, resulting in a poorer customer service than we would have liked and lower output than we might have expected using "modern" production scheduling techniques. Alcan Plate therefore embarked on a path of trying to solve these problems using the very latest techniques and technologies.

The strategy became to make Alcan Plate a computer-controlled plant of the 1990's where automated machine centres would be linked to the main information network for production scheduling and control. The strategy embraced the requirement to implement an on-line manufacturing system integrating a minimum of five sub-systems, namely: sales order handling, shop floor reporting, production scheduling, process control, and testing and release procedures. Figure 1 indicates the inter-relationships of the manufacturing system that we had in mind.

2. The development process

2.1 Design of the overall framework

i) Distributed processing

If the five sub-systems of Alcan Plate's manufacturing system were to operate efficiently, it was clear they would have to be distributed. A two processor system at least would be required to provide a back-up capability, and control systems performing higher speed tasks would need different computers again, located close to the machines they would be controlling. Above all, the requirement for an integrated information system necessitated the computers to talk to one another and called for a distributed data processing (d.d.p.) network.

ii) Selection criteria of the hardware/software

The five sub-systems of Alcan Plate's manufacturing system each had their own requirements per se.

Sales Order Handling was the nearest to a commercial type of application that would be required to be developed, so this would call for a database, screen handling, general report writing and commercial programming language to be available.

Shop Floor Reporting would require special robust, wall mounted terminals with key pads and function keys, supported over long distances (in excess of 1 mile), whereas the system could be written in any suitable language.

The Production Scheduling system would have to access both the sales order handling database, for orders, and the shop floor reporting (or work-in-progress) database for metal in-process and optimise using Operational Research techniques, subject to various production machine constraints. This traditionally called for FORTRAN to be available for this purpose.

In the event of the failure of one processor the other would be required to be used as back up for the systems normally resident on the other machine. This called for an additional, over-riding, requirement: that each of these different types of application programs could be supported on the same machine.

The main supplier to the Alcan Group at this time was Burroughs. They did not have a good FORTRAN compiler, nor did they market shop floor terminals, nor did they support distributed processing.

Computer Museum

DEC and HP were considered. The PDP range of machines was put forward by DEC, specifically the 11/70. Although DEC put forward a number of alternative operating systems none could meet the requirements. Under their multi-purpose Operating System RSX11M they offered TMP70 as a software aid, but this was a non-proven, non-documented database system developed locally by CSS at Reading. The COBOL compiler did not run at that time successfully either. Under TRAX, their database system, FORTRAN was not supported and communication with systems running on other machines was not feasible. In common with HP they did support shop floor terminals and distributed data processing through DECNET. However, the decision finally made itself after a benchmark on the 11/70 indicated no better compile and run times on FORTRAN and much poorer times on COBOL than the HP3000 Series III, and this for a machine which was considerably more expensive.

iii) The approval process

Any subsidiary company of a large Group of companies which has a Central Computer Services Section usually experiences problems with approval to obtain local computing power. In the early 1970's the problems for the subsidiary usually took the form of the Centre persuading the subsidiary that the economics of scale ruled out their request on the basis of cost considerations. This was in general probably good economics and there are plenty of examples of expensive failures of subsidiaries who "did their own thing" as evidence of this. Indeed very close to home, a production control system was developed and installed on a Honeywell computer at Alcan Plate in the late 1960's, and the computer and Department were closed down in 1971! The reasons for their demise were different, but nevertheless, apart from any other problems we would have, we knew we had this little piece of history to overcome!

In the late 1970's the problems for the subsidiary switched to those of incompatibility of hardware with "The Central Machine" or the "prefered" Group hardware, whatever manufacturer this might be. Because of the sheer scope of our systems strategy we probably endured less problems than most. Most of the politics of such decisions revolved around the switching of accounting systems run centrally to more cost-effective in-house machines thereby reducing revenue for the central installation and leaving it with spare capacity (or less of a case for an even bigger new Central machine!). We didn't plan to do this, and so a request to invest £330,000 in computer hardware to develop the integrated manufacturing system, based on a justification of tangible benefits (admin people savings and external time-sharing machine cost savings) and intangible benefits, (a more efficient company) got approval within six months.

For those subsidiaries that have not yet breached the Central control, I forecast that in the 1980's the problems these subsidiary companies will face, will be either

- convincing the Centre that a proliferation of micro-computers linked to one another in a networked multi-user mode can for very low sums of money, perform some of the tasks of the obsolete mainframes (still lurking in the Centre in an undergrowth of inertia and large computer manufacturer politics);
- or convincing the Centre that fourth (never mind fifth yet!) generation software (use of a system generator) does work and you can develop systems ten (or a hundred) times as fast as on the old mainframe through the advantages of prototyping.
- or as fourth generation software becomes available on micros you might have a combination of the two problems (double headache!)

iv) creation and establishment of the 'right' systems team

If you start with no, or very few, people you may well be expected by your management to implement your grand design with the same number of people. Even though you may budget for and include within your request document extra people, and add in realistic sums of money to get them, unless you have an enlightened Personnel Manager/Director you probably won't get any. I didn't! Unless you do get them, you won't succeed and how we eventually got them is a recurring theme throughout the next section.

2.2 The implementation process

i) Systems design

Within systems design is included systems goal or mission, systems requirements (software and hardware), functional specification, and database design. Though prototyping can overcome some of the rigidities of database design, it does not help with the former items and there are no short cuts to success.

A corporate mission listing on one page the main features of the vision that the User (who is he?) has of what the system will do for him, is an essential document to keep referring back to, to check you are still on the right course and that the requirement hasn't changed! The User must be identified at the conception of a project+1 and paradoxically he is not the person who is going to use the system, but is the person who manages other people who will use your system (and these are the "users") when it is completed, whether they like the system or they don't (and often they don't!). Beware of these users, who would, if allowed, direct you to producing a computer replica of the manual system which - because people are cleverer at overcoming undefined exceptions in the manual systems by the use of their brain power and ingenuity - would never work as well as the manual system! And it would need at least as many people to work it, probably more! The User, on the other hand, wants an improvement on the present system and he will need a lot of help from you in order to achieve this objective, because many skeletons (practical obstacles to implementation, both real and imaginery) will appear out of the cupboard.

You will also need a top management sponsor who will sometimes tell the User that he does need what you're telling him he's getting, whenever the User loses faith due to the pressure some of the doubters are applying to him.

All of this applies equally well to non-manufacturing system applications. What makes manufacturing systems more difficult to implement is that the User is likely to be less (or un-)familiar with computer systems and his initial commitment is likely to be low (or non-existent or even anti!). This is because the User will inevitably, and quite rightly, have a production or engineering or metallurgy background, and in

---

1+ If you do not correctly identify the User, your project is likely to fail to be implemented. Alternatively you will waste a lot of time and effort until you find the real User.

traditional British Engineering Industry he is likely to be older. So unlike the Finance people (or wider disciplines in consumer goods industries) he will not have been exposed to computer systems, and words like "on-line" and "database" are likely to be initially confusing to say the least. Furthermore if you do find that your User is an unenlightened uncommitted - "I've been scheduling the mill for the last thirty years and I can't see how it can be done any better" - type, then your top management sponsor needs to be even stronger. Strong enough to fire the man if necessary!

Figure 2 lists the facilities of the Sales Order Handling system which was designed. This list equates with the original systems vision.

The procedure for checking back is as shown in Figure 3. A typical example was as follows:

Some difficulty was encounted through the requirement to implement automatic pricing. The concept is that orders are priced according to structured price lists with discount structures based on customer contracts. An over-ride facility is available once the order has been computer priced, these items appearing on an exception report for control purposes. The problem arises with amendments. Because similar items with the same delivery date are grouped together to create a discount, (part of the structure of the price list), when an item is cancelled or re-scheduled this can cause increases to the original prices of the *other* items. And these amended prices would appear on re-acknowledgements. The problem is that customers have not been used to receiving re-acknowledgements with increased prices. This is because due to the complexity of pricing in the manual system, people more often than not simply 'turned a blind eye'. So the issue hinges on the question "do we want to structure our price lists like this or not?" and not "do we want automatic pricing?".

Manufacturing causes even more emotional issues.

The production scheduling system, as a first stage, allocates orders to produced slabs. Each order planned has a required slab of specific alloy, cross section, length and chemical grade. The production scheduling system automatically matches planned orders and chemically released slabs of the same alloy with regard to certain constraints on cross section, length and grade. Having completed this allocation we have a physically available slab for a planned order of a specific alloy. The allocation system is such that the number of slab-to-order allocations from a current set of orders and slabs is a maximum, i.e. no other combination of slabs and orders can produce more plates against the total order load. In order for the allocation system to be used effectively the slab of specific characteristics must be used against that planned order. The relaxation of this constraint would so de-generate the schedules that certain orders from a specific allocation would not be produced as remaining slabs could not satisfy the remaining order requirements. By taking the allocated slab from one order and placing it against another would cause other slab-to-order allocations to be broken giving delays in their production.

Similarly, a chemically released slab cannot be loaded against a planned order and be sent into production (i.e. begin its rolling program) unless the order has undergone the allocation process and has been allocated a released slab of specific dimensions. Thus if orders are not planned and allocated in advance the released slabs cannot be entered into a rolling program and rolling production could be stopped.

But in a manufacturing mill it was hitherto an unwritten law that the hot-mill would not stop work whilst material was available to roll. Clearly the men working it would otherwise appear to be idle and you would not 'appear' to be very efficient. But theoretically and practically you produce a better downstream schedule if the hot mill is stopped periodically to let sufficient material stack up behind sawing and furnace as rates of production of different products differ across different machine centres. Building up stocks at certain machine centres by planned stoppages of the hot rolling mill allows "smooth" production at machine centres downstream.

Figure 4 and 5 list the goals of our shop floor reporting and production scheduling system and Figure 6 lists the planned and provided system functions.

This section on design would not be complete without a reference to the design team and to the techniques that they would need to employ.

One thing that it is important to stress at this stage, specifically with regard to manufacturing systems in engineering industry, is the need for at least one and ideally more than one member of the team to be fully au fait with the theory and application of O.R. techniques to production scheduling, and optimisation routines. We would otherwise be looking once again at an automated version of the manual scheduling system which is the one we wish to radically improve upon. One of the techniques that was considered and later used was the 'see why' visual simulation package developed by BL. Scheduling rules devised using a full simulation of all major APL machine centres, tested and proven, were later hard coded into the manufacturing system running in FORTRAN on the HP3000 network.

All the other techniques would be those taking advantage of the facilities offered by the HP3000 range of computers.

The Sales Order Handling system would take advantage of VIEW IMAGE and COBOL, plus the ability to mix languages where the automatic pricing could be best written in FORTRAN. The factory link and shop floor reporting would run under MTS to drive the multiplicity of shop floor terminals and screens, and the system would be written in SPL (for efficiency). The data on sales orders would be passed to the work-in-progress database on the one HP3000 using DS3000 from the process planning system (again in COBOL) running on the other HP3000 machine.

Figure 7 shows what hardware configuration we planned and what we would ultimately end up with to support these systems applications.

ii) Project Control

One of the most important aspects of project control is to get the time estimation for the duration of the project somewhere near right. Even more important, whatever timescale you come up with, is to allocate within that timescale appropriate proportions of time for each activity.

Figure 8 shows the breakdown of time between activities for 6 large projects in the US and 2 of Alcan Plate's projects. These indicate that coding always represents a small proportion of total project time and might help explain why most projects take 2 or 3 or 10 times longer than planned. Often people estimate only how long it would take to write the code, and present this as the timescale for the project.

Within the overall project timescale, regular reviews with the User, some of the more enthusiastic of the users, the system authors, and some of the writers, checking that mile-stones have been achieved is essential to timely project success. Simple bar charts are a very effective tool for planning, controlling and measuring project status versus the mile-stones. Signatures of approval, at least for the software requirements and functional specifications are essential. Additionally, quality assurance and training form an essential part of project control. (See the next two sections.)

Many projects based on outside software suppliers tend to flag after the initial enthusiasm to complete the functional specification (and so getting the order!). We employed two software houses initially, since we only had a couple of people, one data processing, one Operational Research. The smaller software house was very helpful but the larger one was totally unsatisfactory. Neither actually completed their respective systems, but the smaller one was high in technical skills and when we eventually took over their work we quickly completed the system, which now does work to a high standard. There is no doubt that without their initial help we would not have been where we are today. The work of the larger company was badly designed and badly written, and very expensive. Effectively the system had to be re-written. The non-timely completion of these two systems brought the issue of in-house versus software house development to a head. Figure 9 is a copy of one of the slides which was used in the presentation aimed at getting top management approval to recruit an in-house team.

The battle over salaries and grades with Personnel over a two year period was finally won and we succeeded in the due course of time, apart from recruiting a handful of programmers, to recruit what a project of this importance needs. And this is senior people, with all the database, data communications and other skills necessary to ensure that you will have 7 day a week, 23 hours a day availability once you have implemented your manufacturing system - because your manufacturing operation soon depends totally upon it. APL managed, in recruiting two such people, to reverse the trend of the drift of "good" people from manufacturing industry to computer manufacturers or software houses. We recruited two senior people from major computer manufacturers! That really does take top management commitment to achieve.

iii) Software Quality Assurance

Very few companies dedicate any resource specifically to control the quality of the systems that their software team complete and hand over to the users. The software houses we used did not do this, and I suspect the majority do not. The major US space and defence projects do include an independent verification and validation, and software quality assurance team to ensure that the system meets its original goals and systems requirements, it is efficient in performance terms (usually defined in the requirements), and that the system is totally "bug" free. Any testing that these people do is also on supposedly "bug" free systems. Obviously in these applications there is a need to get the system "right" first time. Otherwise you end up with dead astronauts and pilots!

The same concepts can and should be applied to the development of all "important" systems. It is the author's personal view that accounting systems cannot have been taken to be very important (except by accountants) as one reason to explain why this function has not existed in traditional dp departments. If the system didn't do what it was originally intended to do, you simply amended it until it did! With a small non-integrated system although undesirable this was possible, but even with a small system it is an expensive way of developing code. With an integrated system it is a prohibitively expensive way of developing code and totally impractical. Manufacturing systems in engineering industry call for integrated and distributed systems, and you have got to get error-free and appropriate modules of code at the earliest possible stage in system development.

The function of software quality assurance in this industry does not call for the same rigorous methodology and numerical staff numbers as the space projects, and a commonsense approach to the problem is all that is required. Someone who can work systematically and logically, checking independendly through user-provided test data the functions of each program as it is developed; the inter-relationships between programs, and who frequently checks the system developed so far back against the original specification, requirements and goal is all that is needed. After initial strained relations between your people you will find that the programmer and designer actually learn to respect this person and look upon it as a challenge to deliver him code which he cannot fault "bug" wise or for its appropriateness.

iv) User education and training

It is important to "sell" the system to the User's peers and colleagues and it is important to get enthusiastic top management support at these sales presentations. Obviously the system is sold under the guise of user education, and the assembled gathering do learn a great deal about the scope, nature and benefits of the system which is about to be implemented. The objective of these sessions is to gain commitment by all those who will directly or indirectly manage the people who will actually be asked to use the system, so that they will support the system when it is "knocked" and may, through this training, even be able to explain the issue under attack, rather than passively agree with the person making it.

It is also essential to properly train those who have to use the system long before they have to use it in a "live" situation.

Sufficient time should be set aside for this activity, and under no circumstances should the trainer be the system designer or programmer. Nor should the training be done with a system full of unfound "bugs" since this rapidly generates lack of credibility and user disinterest in the system. It is however acceptable to use for such training a system which does not yet have all the available features or may not have been tested down a fully exhaustive set of program paths. Finding the odd "bug" at this stage is not unhelpful and obviously it is better than finding them when you are "live". In this phase of training although there is some selling involved, there is much less emphasis on this aspect and much more on a thorough mundane detailed step-by-step "this is how you do the following", and "this is what you do next".

v) Systems enhancements and maintenance

These are to be avoided.

The system you have implemented meets the goals and system requirements and conforms to the original specification. The users have been properly trained and the User and his colleagues are knowledgeable and committed to what you have developed. The system is also very nearly "bug" free.

There are always teething problems. If you have done a good job 89 per cent of the difficulties will, even despite your thorough training, be user unfamiliarity with some of the more obscure system features, a lack of knowledge of how to get out of, or into, or use some of the less frequently called-for facilities. Your systems' trainer will, together with the system designer, sort these problems out quickly as they occur. 9 per cent of the difficulties will (despite your quality assurance techniques) be system "bugs". If they aren't you've done too much testing before you went "live" and probably the system isn't now needed any more! Your designer and programming team will readily fix these.

The remaining two-per-cent will be system features which the users cannot manage without. If the User agrees, and you have exhausted the use of the other faciltiies you have provided, and you can't persuade them to live without this new facility then, and only then, should you agree to an enhancement. And then "stall for time", because the requirement might still go away! This is because the users will increasingly get happier and happier as they become more familiar with the new system and will start to enjoy its good features rather than look for real or perceived limitations.

3. The advantages of a computer aided manufacturing system in an engineering environment

At Alcan Plate the effects on production flow and throughput have been quite staggering. The inflexibility which is inherent in a (and our) manufacturing system is no longer referred to (not even by our Sales Department!) because the efficiency of production and the resultant output is so much improved. Customer service both through the improved information flow and the efficiency of the system per se is significantly better.

In general the benefits provided by the implementation of such a manufacturing system are as summarised in Figure 10.

For a company which has it in mind to develop a manufacturing system in such an environment, I would like to finish with two slides indicating what I believe are some further problems which have not been touched on earlier, in Figure 11, and - if you are still interested! - to provide you with a summary of some useful tips, and these are shown in Figure 12. May I wish you the best of luck, you will need a lot of that as well!

Figure 1

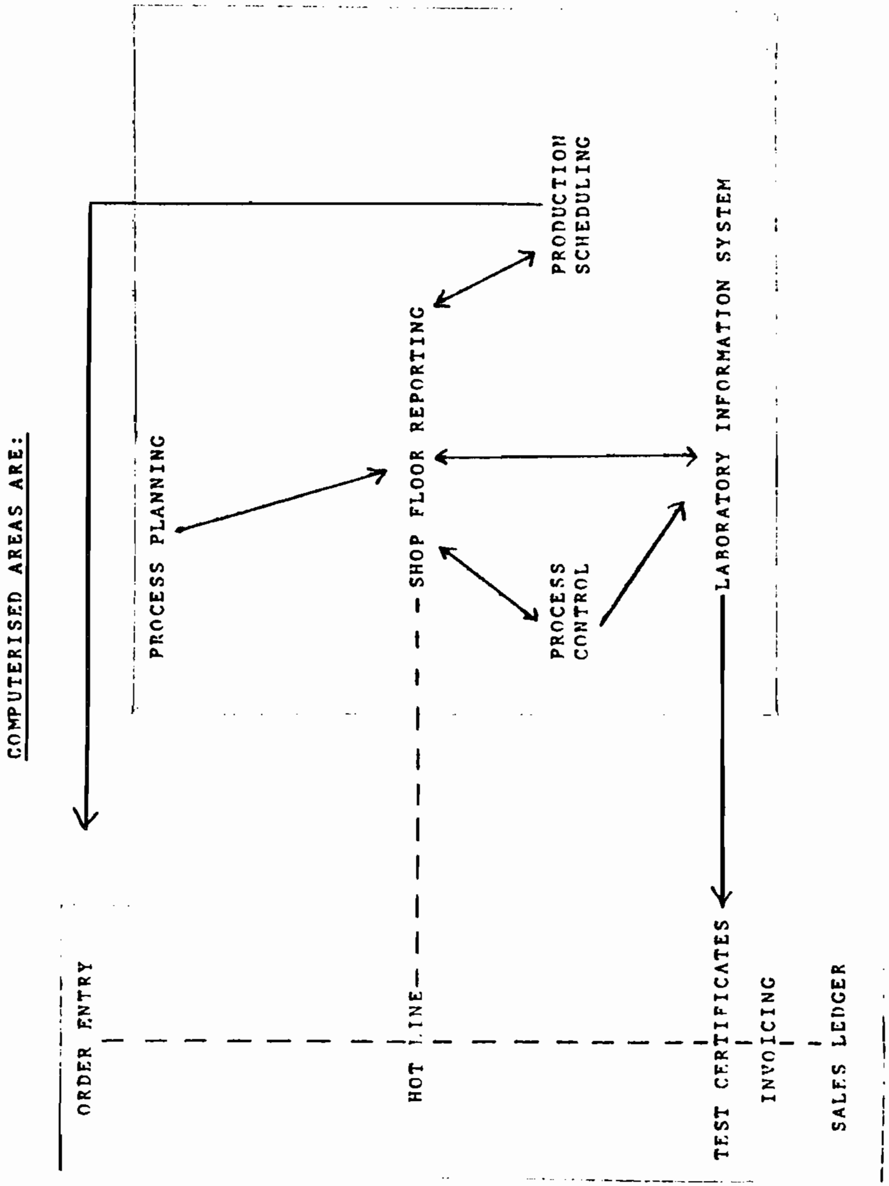

COMPUTERISED AREAS ARE:
PROCESS PLANNING
SHOP FLOOR REPORTING
PRODUCTION SCHEDULING
PROCESS CONTROL
LABORATORY INFORMATION SYSTEM
ORDER ENTRY
HOT LINE
TEST CERTIFICATES
INVOICING
SALES LEDGER

Figure 2

SALES ORDER HANDLING FACILITITES INCLUDE:

1. ORDER ENTRY

2. AUTOMATIC PRICING (USING PRICE LISTS AND CUSTOMER CONTRACTS, SPM's)

3. ORDER ACKNOWLEDGEMENT

4. ORDER AMENDMENT, RE-ACKNOWLEDGEMENT (+ RE-PRICING).

5. FORWARD EXCHANGE COVER OF NEW BUSINESS.

6. AUTOMATIC ROUTE PLANNING AND MILL CARD PRODUCTION.

7. ENTRY OF PACKING RECORDS.

8. PRODUCTION OF ADVICE NOTES, RELEASE NOTES AND TEST CERTIFICATES.

9. PRODUCTION OF INVOICES + BANBURY SALES LEDGER TAPE.

10. COMPREHENSIVE ENQUIRY FACILITIES.

11. SALES LOADING.

12. COMPREHENSIVE SYSTEM OF MANAGEMENT INFORMATION REPORTS. (E.G. BOOKINGS + PRICE PERFORMANCE VS. COMPANY PLAN)

13. PRODUCTION OF CREDIT NOTES.

Figure 3

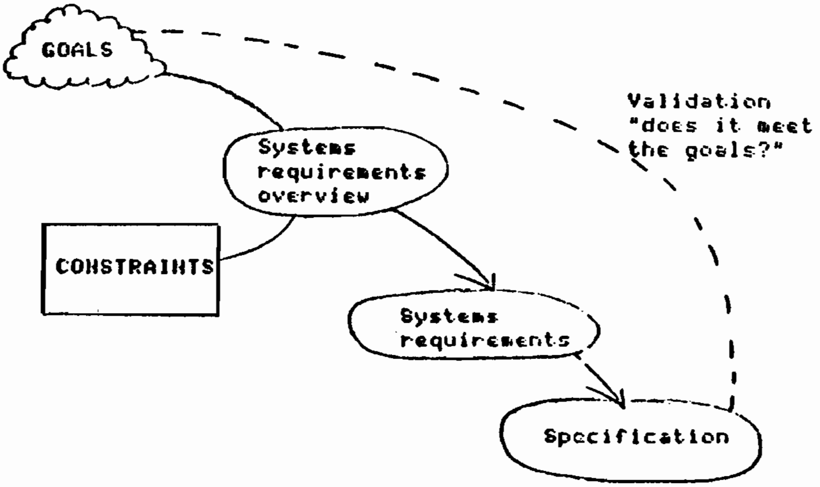

GOALS
Validation
"does it meet
the goals?"
Systems
requirements
overview
CONSTRAINTS
Systems
requirements
Specification

Figure 4

GOALS

SHOP FLOOR REPORTING

REPLACEMENT OF LOG SHEETS.

PAPERLESS SOCIETY (MILL CARDS KEPT IN SALES ADMIN. AS A BACK-UP CONTINGENCY).

TODAY'S WORK SCHEDULE DISPLAYED AT EACH MACHINE CENTRE.

WORK DETAILS OF NEXT JOB DISPLAYED AT EACH MACHINE CENTRE (E.G. CUT TO THIS SIZE).

(AUTOMATIC DATA MEASUREMENT WHERE POSSIBLE TO BE INTEGRATED).

A TRACKING SYSTEM (MANAGEMENT INFORMATION SYSTEM).

Figure 5

PRODUCTION SCHEDULING

TO PRODUCE SCHEDULES USING UP-TO-DATE INFORMATION SCHEDULES FOR EACH MACHINE CENTRE.

IMPROVE PRODUCTION FLOW.

LOWER W.I.P.

INCREASE FACTORY UTILISATION.

INCREASE EFFICIENT PRODUCTION.

INCREASE PRODUCTION CAPACITY.

Figure 6

INTEGRATED MANUFACTURING SYSTEM FUNCTIONS INCLUDE:-

1. RECORDING OF OPERATOR'S IDENTITY.

2. COMPREHENSIVE REPORTING OF SLAB DATA AT MACHINE CENTRES.

3. CHEMICAL LABORATORY HOLD/RESTRICT/RELEASE OF SLABS.

4. VALIDATION OF SLAB DATA ENTERED AT MACHINE CENTRES.

5. UP-TO-DATE SLAB INVENTORY.

6. AUTOMATIC TRANSFER OF MILLCARDS FROM PROCESS PLANNING WITH GENERATION OF PROCESS PLATE NUMBERS.

7. AUTOMATIC SLAB TO MILLCARD ALLOCATION OF RELEASED SLABS.

8. ALLOCATION OF RESTRICTED SLABS TO ORDERS.

9. AUTOMATIC SCHEDULING FOR THE HHT INCLUDING DETERMINATION OF THE OPTIMUM HOTROLL ORDER, HHT SEQUENCE, PALLET AND SAWBED POSITIONS OF PROCESS PLATES.

10. AUTOMATIC UPDATING OF SHOP FLOOR REPORTING DATABASE WITH GENERATED SCHEDULING INFORMATION.

11. AUTOMATIC PRODUCTION OF HOTMILL LOG SHEETS AND HOT LINE PRODUCTION REPORTS.

12. COMPREHENSIVE SYSTEM OF REPORTS, E.G. SLABS AND PLATES SCRAPPED, OVERDUE, ETC.

13. AUTOMATIC ENVELOPING OF PLATES FOR HHT AND TRANSFER OF DATA WITH THE SIEMENS FURNACE CONTROL SYSTEM.

14. ENQUIRY FACILITIES TO IMPROVE TRACKING OF PLATES.

15. AUTOMATIC ARCHIVING OF SLAB DATA.

Alcan Plate Ltd. Hardware and Systems Configuration 18th. March 1982

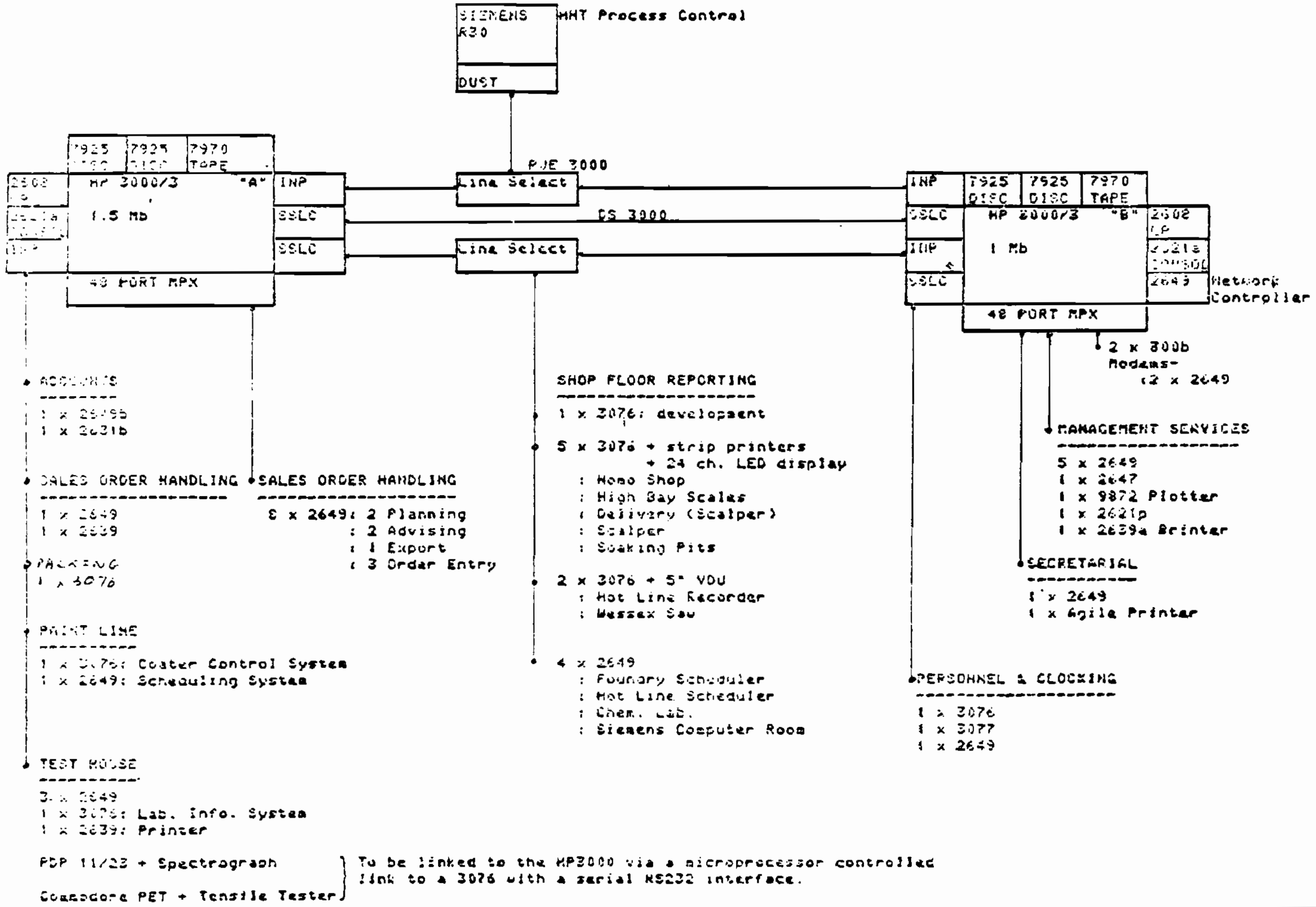

SIEMENS R30
HHT Process Control
DUST
RJE 3000
Line Select
DS 3000
Line Select
7925 DISC
7925 DISC
7970 TAPE
2608 LP
2621a CONSOL
DUP
HP 3000/3 "A"
1.5 Mb
48 PORT MPX
INP
SSLC
SSLC
INP
SSLC
IOP
SSLC
7925 DISC
7925 DISC
7970 TAPE
HP 3000/3 "B"
1 Mb
48 PORT MPX
2608 LP
2621a CONSOL
2649
Network Controller
2 x 300b Modems-
(2 x 2649
ACCOUNTS
1 x 2649b
1 x 2631b
SALES ORDER HANDLING
1 x 2649
1 x 2639
PACKING
1 x 3076
PRINT LINE
1 x 3076: Coater Control System
1 x 2649: Scheduling System
TEST HOUSE
3 x 2649
1 x 3076: Lab. Info. System
1 x 2639: Printer
PDP 11/23 + Spectrograph
Commodore PET + Tensile Tester
To be linked to the HP3000 via a microprocessor controlled link to a 3076 with a serial RS232 interface.
SALES ORDER HANDLING
8 x 2649: 2 Planning
: 2 Advising
: 1 Export
: 3 Order Entry
SHOP FLOOR REPORTING
1 x 3076: development
5 x 3076 + strip printers
+ 24 ch. LED display
: Homo Shop
: High Bay Scales
: Delivery (Scalper)
: Scalper
: Soaking Pits
2 x 3076 + 5" VDU
: Hot Line Recorder
: Wessex Saw
4 x 2649
: Foundry Scheduler
: Hot Line Scheduler
: Chem. Lab.
: Siemens Computer Room
MANAGEMENT SERVICES
5 x 2649
1 x 2647
1 x 9872 Plotter
1 x 2621p
1 x 2639a Printer
SECRETARIAL
1 x 2649
1 x Agile Printer
PERSONNEL & CLOCKING
1 x 3076
1 x 3077
1 x 2649

Figure 7

Figure 8

SEVEN EXAMPLES OF TIME CONSUMPTION
(ALL EXCLUDE FORMAL DOCUMENTATION)

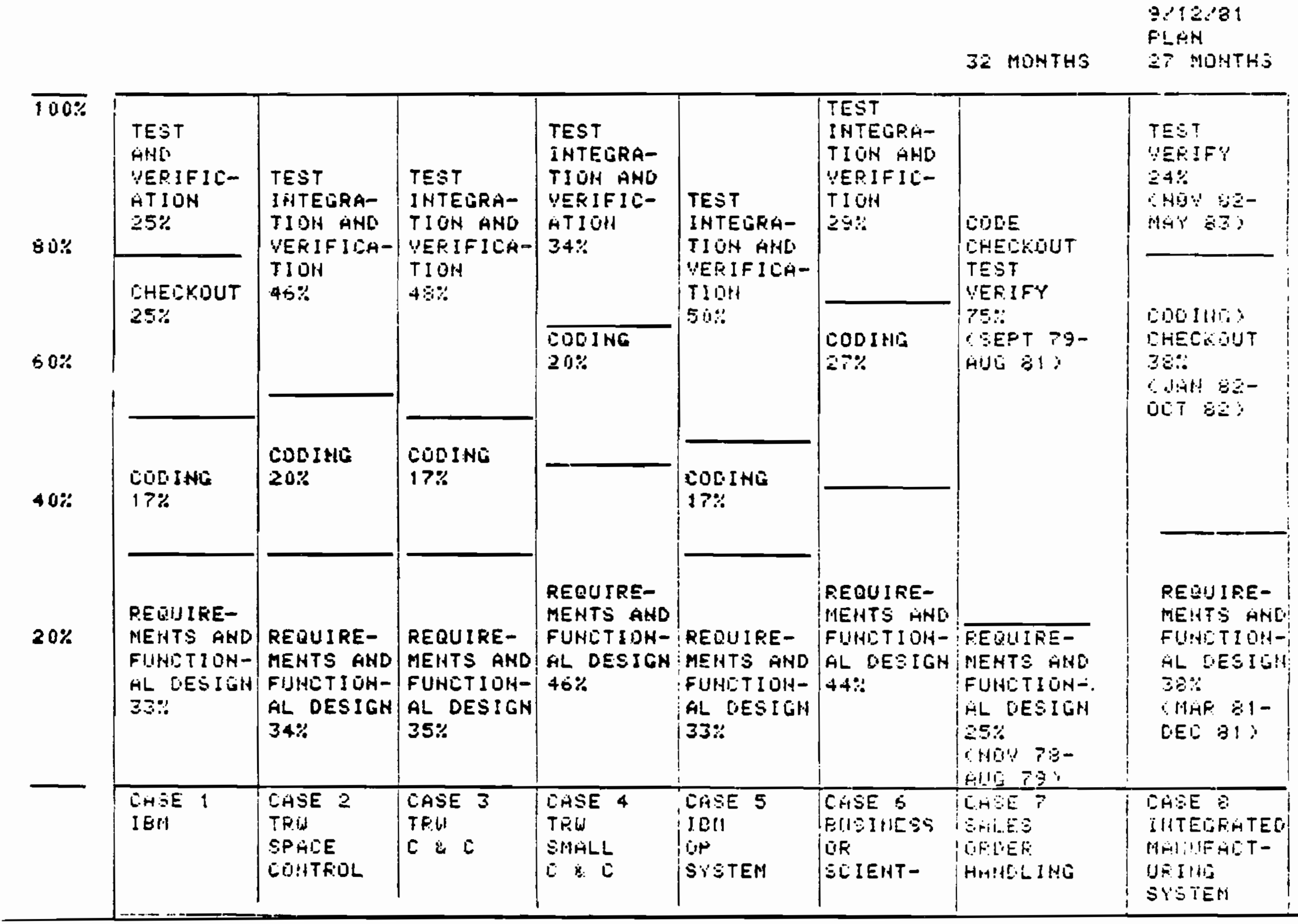


Figure 9

WAYS CONSIDERED TO GO FORWARD

1. OUTSIDE SOFTWARE SUPPLIERS

   - EASY TO DO
   - EXPENSIVE
   - DIFFICULT TO CONTROL
   - NOT IN THE BUSINESS OF 'MAINTAINING' SYSTEMS
   - INSUFFICIENT KNOWLEDGE OF OUR BUSINESS

2. IN-HOUSE SOFTWARE EXPERTISE

   - MINIMUM COST ROUTE
   - MANAGEABLE
   - MORE EFFECTIVE MAINTENANCE OF SYSTEMS
   - TAKES LONGER
   - BEWARE OF 'SOFTWARE WIZARDS'

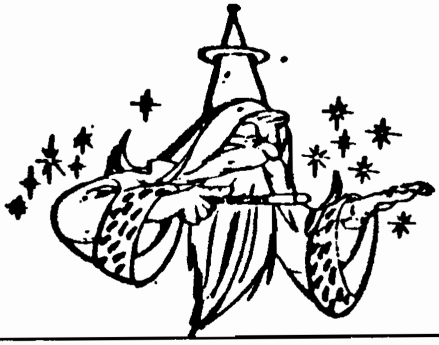


Figure 10

COMPUTER AIDS TO MANUFACTURING

ADVANTAGES OF AN INTEGRATED MANUFACTURING SYSTEM

- PROVIDES UP-TO-DATE MANAGEMENT INFORMATION, AND IS THEREFORE THE BASIS FOR THE CONTROL OF THE BUSINESS. (ORDER INTAKE, PRICE LEVELS, ETC.)
- PERMITS SCIENTIFIC PRODUCTION SCHEDULING (BENEFITS OF EFFICIENT PRODUCTION, INCREASED PRODUCTIVE CAPACITY, ETC.)
- PERMITS PRODUCTION CONTROL
- IS NOT LABOUR-INTENSIVE
- YOU BECOME A COMPETITIVE COMPANY

Figure 11

COMPUTER AIDS TO MANUFACTURING

PROBLEMS

- THE COST (IS "EXPENSIVE")
- THE TIME (TAKES A "LONG" TIME)
- PIONEERING (IF YOU'RE NOT THEN YOUR ADVANTAGE IS LOST)
- "EMPIRE BUILDING" AT THE EXPENSE OF OTHER "FUNCTION'S PATCHES"

Figure 12

COMPUTER AIDS TO MANUFACTURING

TIPS

- BEWARE OF SOFTWARE WIZARDS (IF YOU CAN FIND THEM)

- BUT BE EVEN MORE BEWARE OF
  - COMPUTER MANUFACTURERS (DP MEN GET HUNG UP ON REPLACEMENT HARDWARE/RENTAL, ETC.)
  - SOFTWARE HOUSES, PARTICULARLY THE FAMOUS NAMES (THERE ARE PLENTY OF THESE)

- DON'T TAKE SHORT CUTS (I.E. NO SYSTEM SPECIFICATION, NO TEST PROCEDURES) - THERE AREN'T ANY.

- DO KEEP CHECKING BACK (OFTEN) AGAINST THE CORPORATE MISSION, GOAL, PLAN. (BECAUSE OF THE TIME DEVELOPMENT TAKES, THE BUSINESS HAS OFTEN CHANGED DIRECTION IN THE MEANTIME!)

Nigel Fielden
Principal Consultant
Hoskyns Group

HOSKYNS INTEGRATED PURCHASING SYSTEM

1. INTRODUCTION

Hoskyns Integrated Purchasing System (IPS) is a powerful purchasing and materials management system which can be easily customised to suit the needs of any organisation.

IPS is a self-contained system which interfaces directly with HP's Material Management system (MM/3000). The system is equally suited to both new and existing users of MM/3000. Interfaces are available through IPS into other systems such as Accounts Payable and Costing. IPS forms a convenient building block for the user as it requires no changes to MM/3000 and can be quickly installed.

The user of IPS has access to a powerful, user friendly purchasing system, linked into MM/3000, which is itself a proven application. The screen entry facilities are compatible with those provided by MM/3000 and are extremely user friendly. The system is completely menu driven with function keys used to guide the user through the system.

The link between MM/3000 and IPS is achieved by means of a standard interface. Transactions which pass into MM/3000 are fully compatible with the processing rules defined within MM/3000.

MRP can be used to automatically produce new purchase orders or to provide a suggested order list which can be manually overridden. Full purchase order tracking facilities are available within IPS with receipts against orders automatically updating stock figures.

Invoices entered into IPS are checked against order and receipt details and, if approved for payment, by the system are passed through to the Accounts Payable interface.

2. ELEMENTS OF IPS

IPS consists of the following elements:

- materials catalogue with complete buying information complementing the material information held in MM.
- supplier details - these may be maintained within IPS or accessed from a bought ledger system.
- pricing information - comprehensive details on a supplier per materials basis.
- full order administration ie.
    - order processing
    - receipts/returns
    - invoices/credits
- comprehensive on-line rapid response enquiries providing immediate access to up-to-date information.
- user controlled reporting system with full scheduling and automatic rescheduling of all batch jobs within the system.
- interfaces into other systems (eg. accounting, costing etc.)

A check list of features is shown overleaf.

CHECKLIST OF FEATURES

BUYING INFORMATION

- Materials Catalogue
- Family groupings of materials
- Material File Lists
- Supplier File
- Supplier Intelligence Information
- Supplier Lists
- Price Details
- Supplier/Material Linking
- Automatic Price Updating
- Materials by Supplier List/Enquiry
- Order Status Enquiry
- Outstanding Orders by Supplier Enquiry
- Outstanding Orders by Material Enquiry
- Order History Enquiry (by Supplier, Order)
- Invoice History Enquiry (by Value, Order, Supplier or Invoice Reference).

ORDER ADMINISTRATION

- Automatic Orders
- Manual Stockable Orders
- Text Processing
- Order Amendments
- Order Acknowledgements
- Confirmation Orders
- Order Print
- Multi-item Orders
- Goods Receipts Entry
- Supplier Returns
- Invoice Entry (Orders in System)
- Invoice Batching/Reconciliation
- Automatic Approval of Invoices
- Payment Stops (by order or supplier)
- Purchase Ledger Interface

MANAGEMENT INFORMATION

- Order Status Enquiry
- Order History Enquiry
- Invoice History Enquiry
- GRNI
- Commitments
- Invoice Blacklist
- Daily Batch Report
- Performance Statistics
- Comprehensive Expediting/Project Enquiry System
- Supplier Performance
- Invoice Price Variations
- History Analyses
- Inflation Report
- Costing Report

CONTROL DATABASE

- Group structure and company profiles, addresses etc.
- Authorised users
- Standard Financial parameters - e.g. VAT and currency rates
- Company calendar information
- Error messages, field names and report headings
- Payment terms
- Invoice Approval criteria
- Product groupings
- Standard clauses etc.

3. GENERAL ATTRIBUTES

IPS is constructed to allow the user to implement the various elements in stages. The Control Database sub-system enables the user to tailor the functionality of the system to meet specific requirements. The interactive transaction processing facilities together with the fully integrated database ensure the availability of timely and up-to-date management information.

Key attributes of IPS are:-

- User specified control parameters
- On-line input of transactions and maintenance data
- Interactive update
- On-line enquiry to current and historical information
- Full audit trails and recovery facilities
- Full security controls
- Flexible processing schedules
- Ability to interface with other systems
- Ease of maintenance

4. DESCRIPTION OF IPS

IPS consists of the following modules:

- Material Catalogue
- Supplier Details
- Pricing Information
- Full Order Administration
- Comprehensive on-line enquiries
- User Controlled Reporting
- Interfaces to other systems

4.1 Material Catalogue

Materials within a purchasing system will usually fall into one of three categories:

- a standard item, which is purchased regularly and for which demands are met from stock.

- a standard item which is purchased regularly but not stocked - supplies when received going directly to the end user.

- a non-standard item or service.

The materials catalogue in IPS is used to handle all buying information for materials in the first two categories. Items may be stocked at multiple locations. IPS also allows the materials defined in the catalogue to be grouped into "families" since much of the descriptive information for one material in the system may be common to several other materials. An additional benefit from using this structure is that classified lists of materials stocked are easier to produce.

For the third category of material, the descriptive information is usually obtained from the puchase requisition document and is entered directly into the order.

4.2 Supplier Details

IPS provides a supplier file maintenance facility which can be restricted to authorised personnel. Information held on the supplier file, relating directly to purchasing includes:

- minimum order value (if imposed)
- payment terms
- conditions of purchase agreement number
- supplier name and address
- cumulative value totals, both for orders and invoices, year to date and last year
- carriage charges are monitored, together with the total number of invoices.

In extreme circumstances, all payments may be stopped to a supplier, or invoices for a particular supplier may be prevented from being automaticaly approved by the system. A scratch pad area is provided for the buyer to record useful information about the supplier - retrieval of this information is on-line.

4.3 Pricing Information

Pricing information is held in IPS as one record for each supplier/ material combination. Each material in the catalogue may have an unlimited number of price records against it. Individual suppliers may be nominated as the preferred supplier for a material at either individual locations or across all locations.

Information held on the price record includes current price, discount structure, contract reference number (if appropriate), carriage information and locations served by this supplier.

IPS automatically maintains expenditure information for each supplier per material record. The expenditure is held as quantity and value totals for the year-to-date, last year and the previous year. The information on the price file is available on demand through a rapid response enquiry.

4.4 Order Administration

This section describes how orders are raised and printed, and how subsequent amendments, receipts and invoices are handled.

4.4.1 Text Processing

One convenient feature of the system is the ability to retrieve standard paragraphs of text, and incorporate these into the purchase order on printing. All such paragraphs are retrieved from tables, so changes to wording are very easily accomplished.

Each paragraph is called up by using a short mnemonic code.

4.4.2 Types of Order

There are three basic methods of raising orders:

- for stocked items (which are not automatically re-ordered) the purchase order is initiated through a "stock review" procedure. Since the bulk of the information has been pre-set by the user, only a simple confirmation of the order content is required in most instances.

- for stocked items which are automatically ordered IPS will raise orders on the preferred supplier based on suggested orders from MRP. These automatic orders are summarised on a report which is normally produced at the start of each day. Prior to the order print run, the user may amend or cancel any of the automatic orders - the orders must then be released on a factory basis before the modified orders are printed.

- for non-stock items, the information has to be captured from the indent or purchase requisition. There is necessarily a significant amount of data to be entered, and this tends to be carried out by typists using visual display units.

  Once entered, both types of order are treated identically, and appear in the same form to the supplier.

4.4.3 Stocked Item Review

The review of an item suggested for ordering by MRP is carried out interactively by the user at his terminal. The steps are as follows:

- he identifies the material, and verifies the description which is displayed to him

- he approves the supplier to him, or selects an alternative from those held on the price file.

- control figures are also immediately recalculated and displayed, including Economic Order Quantity, Re-order level, Safety stock and lead time.

- he determines and enters the order quantity, and approves (or amends) the delivery date; he can also add any special instructions for the supplier.

- finally he enters the order number, or allows the system to assign an order number and provided all the information is correct, releases the order for printing.

The whole of this process can be accomplished with a minimum of keying.

Facilities are provided for confirmation orders, for multi-item orders, and for "local stockist" or weekly orders. In the latter case, the order is not printed immediately, but held open for a week - further items will be added throughout the week and a large multi-item order generated at a pre-set time. This minimises the amount of paper generated for low-value items.

4.4.4 Non-Stock Item - Order Entry

As mentioned above, this tends to be a high speed data entry function. The indent forms the source document, has information added to it by the Buyer before entry, identifying supplier, price and so on.

The order consists of a heading, item lines, cost allocations and buying analysis information. Up to 99 items are allowed per order, and individual lines may have split cost allocations. The buying analysis is used for later information retrieval from the Order History file.

It is possible to call up a standard description by quoting a material code, even if this is not a stocked item. Similarly, standard paragraphs may be called up and embedded within the order description.

4.4.5 Order Amendments/Acknowledgements

Changes to existing orders are usually processed directly, the system will display the Order on the terminal and the required fields are changed. Virtually any field of information may be changed, subject to security clearance.

A special high-speed routine is provided for acknowledgements - this takes the user through only those fields likely to be changed - suppliers' reference, delivery date and price.

On completion of a change, the user can instruct the system to print a formal amendment order, to reprint the order from scratch, or to suppress printing.

One special form of amendment is order closing, which prevents any further transactions (receipts, invoices) against an order. For security reasons the order is not removed from file immediately.

4.4.6 Order Printing

A "same day' service is provided on order printing - an order raised anywhere in the system will be printed and in the post to the supplier on the same day.

Amendment orders are produced on the same order stationery.

Multi-page orders are provided, with continuation facilities; full stationery line-up and re-start facilities are also provided.

4.4.7 Goods Receipts

Entry of goods received is a high-speed process; immediate verification of the information takes place against the Purchase Order held on file. Multiple deliveries against an order item are allowed and tracked, and it is alsopossible to accept and hold an invoice pending the receipt of goods; returns to supplier are processed.

If this is the first receipt of this item against this order, then the purchasing lead time on the stock record for that item is automatically updated.

4.4.8 Invoice Entry

When the invoice is received from the Supplier, it is immediately registered within the computer system. For control purposes, this is done in batches of 20-30 documents and control totals kept. If the invoice balances within itself, and if it matches the quantity received it is tested for automatic payment approval. This can take place if the invoiced price matches the order price within criteria specified by the user.

If automatic approval cannot be given, the invoice has to go to the Buyer (and possibly the originator) for manual approval.

Invoices can be accepted which do not match orders in the system - for example for electricity, gas etc. This provides for comprehensive control reporting and analysis of all purchases. These invoices may also be registered prior to approval being obtained.

For security reasons, cumulative totals are kept by supplier to show any "drift" between order prices and invoice prices. If the user feels that a Supplier is taking advantage of the automatic approval process, he can put an immediate "stop" on that supplier, after which all invoices must be manually approved.

For payment purposes an "interface" file is generated for transfer to the Accounts Payable System.

4.4.9 Order History

Facilities are provided for "off peak" enquiries into the Order History which may be very large in volume. The enquiries can specify any combination of;

- location (site)
- order number
- product group
- supplier
- range of dates
- material code
- noun
- type
- material type
- purchase analysis code

The system will search the order history, remove details of all order lines satisfying the combination of criteria, and provide a printed analysis.

4.5 Comprehensive on-line enquiries

In addition to those facilities within MM/3000, IPS also provides enquiries on:

- materials
- suppliers
- prices
- order status by material, supplier or order number
- order history for completed orders
- invoices awaiting approval
- expenditure by material or by supplier
- invoice history
- expediting status

4.6 User Controlled Reporting

IPS provides several reports - these may be run at regular intervals or on demand. A list of these reports is provided in the checklist of features section.

In order to aid the user in scheduling reports, IPS provides a complete job scheduler which can handle any job in the system on either a regular or ad-hoc basis. For those jobs within IPS that require parameter input, the scheduler offers on-line validation of the parameters. Jobs may be scheduled to run daily, weekly, monthly or on any regular cycle. The schedule may be amended by the user and the system provides comprehensive information on the current status of the schedule.

4.7 Interfaces to other systems

The system provides an interface into Accounts Payable as standard. This interface passes details of invoice payments and costs. Interfaces are also available to pass accruals and stock consumptions into the General Ledger.

Due to the system's modularity it is a simple task to build further interfaces into the system. These would include interfacing with sales order processing and job costing.

Getting Started in Data Capture

Bruce Toback
Infotek Systems
1400 N. Baxter St.
Anaheim, CA 92806
U.S.A.

I. Introduction

Over the past decade, on-line computer usage has been replacing traditional batch usage at an increasing rate. The reasons for this transformation are many, but most center around responsiveness: the ability to use the computer as an information source in real time, asking questions which require the most current answers.

In point of fact, however, a large portion of this transformation has been the replacement of batch systems with "faster" batch systems. Data are still collected and recorded manually, then sent to a terminal operator for entry into the "on-line" system. This method has advantages over traditional batch systems in that data are made available sooner than with conventional batch processing and inquiry is easier, but responsiveness can still suffer. An additional consideration is the accuracy of information presented to the computer. In both traditional batch systems and "on-line" batch systems, information is copied at least twice and perhaps several times on its way to the computer. Each translation increases the chance for error.

More recently, source data capture has been introduced. The idea of source data capture is not new, of course: every mechanical cash register is a source data capture facility, recording sales transaction information as each transaction occurs. Using source data capture as input to a computer, however, is a relatively recent innovation. Data capture has been in use quite extensively in the retail sales field in the form of point-of-sale terminals and is now being used in manufacturing as well.

This paper is intended as an introduction to data capture. It will serve as a guide during a data capture project definition, and provide a list of ideas and issues to focus on as you design your data capture system.

II. When data capture is used

In general, data capture is useful in a manufacturing environment when at least one of the following is true:

- o The information needed is highly volatile (i.e., changes frequently)
- o The information needed is too voluminous to be recorded and keypunched in a timely manner
- o The information required is already available in machine readable form as a by-product of an existing manufacturing operation.

Note that in general, source data capture means only that information is recorded electronically as soon as it is generated. This may mean that electronically-assisted means exist to capture the requrired information or may simply mean that the person creating the information (e.g., the sales clerk or stock clerk) enters the information by hand into a suitable terminal.

Let us examine each of these criteria in detail. First, information may be considered volatile when its lifetime is shorter than the turn-around time of the system used to record it. For example, on a fast-moving shop floor, a given assembly may move in processing from operation to operation several times in one day. If information is entered and recorded once a day, the report will be accurate only for those assemblies that are stuck. This kind of reporting is usually of limited value.

Information which is too voluminous to be keypunched normally can include such things as point-of-sale inventory transactions, e.g., grocery checkout; kit pull information when lot tracking is in use; and of course, job tracking information on a busy shop floor. In general, the volatility of information is related to its volume: information which consists of only a few data items can usually be entered into a data processing system in time to be reported before the end of its useful life.

Finally, information which may already be available in machine readable form includes such things as, again, grocery checkout, where UPC or EAN codes are pre-stamped on packages by the manufacturer; electronic time clocks which can record time and attendance information at the same time that an employee's time card is being punched; and electronic test equipment which may be computer controlled. In some cases, as in retail checkout, the machine-readable coding may be inherent in the operation being tracked by the computer. In other cases, such coding may be added at neglegible cost, such as when turn-around documents are already produced by computer.

III. Defining your Data Capture Needs

Once you have determined that your data entry needs can be met by a data capture system, your next task will be to design the system. The design can be broken down into three main concerns:

- o Volume capabilities
- o Volatility problems
- o Using preexisting data

Volume Capabilities

In order to design a data capture system, an upper limit on transaction volume must be determined. How this is done depends on the type of data that you intend to capture, and on what capabilities already exist for capturing it. When converting from an "on-line batch" system, the best indicator may be the number of turn-around documents currently being keyed into the system, multiplied by the number of transactions recorded on each document. This last is very important since in a properly designed data capture system, each "real" transaction (e.g., item sold, job move, etc.) should equate to one computer transaction.

If no automated data processing facility is currently in use to record the information you are trying to capture, then the volume should be estimated by the personnel most involved with the data.

Number of terminals Once you have determined the total transaction volume for each function you wish to perform, you will need to find the number of terminals required to support that volume. The number of terminals required depends both on the total transaction volume and the number of types of transactions.

Transaction types are important becase each function may take a different amount of time to perform, and more importantly, may take place at a different location from other functions. In general, enough terminals to support the volume for a particular function should be present in the immediate area of transaction generation. This is self-evident when machine-readable information is input through the terminals, but may not be so evident when operators are required to key in information. If an operator needs to "go over to" a terminal to make an input, the tendency will be for the operator to batch input and do several transactions at one time. This may result in both translation problems as operators write down transactions for later entry, or in timing problems as operators delay entering important information until a break period.

For hand data entry, approximately five seconds are required for a 10-digit numeric entry. This may be reduced substantially by using a machine-readable code such as bar code or OCR.

Machine-readable Input Machine-readable input can have a significant effect on the volume capabilities and terminal requirements of your data capture system. First, of course, data entry speed can be increased dramatically because operator keying is no longer required. Bar code or OCR scanning can reduce the operation of entering a 10-digit work order number to a single wand stroke, with almost no possibility of keying errors. However, a "one-operator, one-station" arrangement becomes almost mandatory, since sharing a "personal" device such as a bar code wand is generally reported as difficult. If information is coming from another computer, as in the case of automatic test result reporting, the system must be *by definition* one-operator, one station.

Volatility Problems

If the data you are trying to capture is highly volatile, you will need to consider several additional factors. Most of these are treated as application design problems, but will need to be considered during your system definition.

System availability Availability of the computer system is of interest in even a batch system, but is much more important in an on-line system. Availability is critical to the success of a data capture system dealing with volatile data. Imagine for a moment the chaos that might result from a malfunction of a centralized electronic door lock controll The data in this case are extremely volatile – someone wants in (or out) *now* – and it is gone forever if it is not captured at the instant it is generated. If your data are this volatile, you should make provision for spare or backup terminals, and for rapid switchover to a backup computer if practical.

In a situation involving volatile data, terminal reliability is as important as central system reliability. If an operator loses access to the system even though the system is still operational, data from that operator will be batched or lost, or the operator will simply not do any work until access is restored.

Many of the schemes for connecting data capture terminals to the HP3000 involve shared equipment, leading to the possibility of single-point system failure. For example, if all terminals are on a single multipoint line, failure of the INP or SSLC, of the factory data link adaptor (3074A), the cable itself, or certain terminal failures can render the entire data capture network inoperable. Suitable backup precautions must be taken, e.g., splitting the network into two or more lines, or connecting critical terminals point-to-point so that failure of an ATP, ATC, or ADCC can be worked around rapidly by moving the terminal to a port on a different ATP, ATC, or ADCC. The net effect of these considerations is that you should consider the possibility of using redundant hardware in situations in which data are extremely volatile.

Data Recovery In addition to high data collection equipment uptime, volatile data collection requires that you consider recovery methodologies in the event of system failures. These can take several forms. Assuming that retention of paper backup is undesirable, or that no paper backup exists to begin with, the most obvious recovery technique is user logging. This service, provided at an elementary level by MPE and at a more sophisticated level by IMAGE, causes MPE to attempt to journalize every file update to tape. The implication, however, is that you are willing to dedicate a serial medium, either tape, disc, or cartridge tape, to data logging whenever the data capture system is active. You should plan for this in determining your hardware requirements.

In addition to hardware requirements for journalization, you will need to examine your backup strategy for compatibility with logging. In particular, if you are using IMAGE, you *should* use DBSTORE instead of STORE or SYSDUMP to back up. This is because the Hewlett-Packard logging/recovery utility, DBRECOV, will examine the last *DBSTORE* date to determine whether your transaction log and data base "match" for recovery. If you normally do long-period incremental dumps (i.e., every two or three days), you may in addition be faced with a considerable recovery period after a system crash. While this is preferable to losing several days' worth of transactions (!), it also causes a *net* decrease in system availability as perceived by your data capture operators.

Finally, if you are not using IMAGE, you will need to design your own recovery utilities for your data base. If you implement your data base with KSAM or with a third-party data base system, this may be an important planning consideration.

Machine-readable documents If you choose to use paper backup instead of transaction logging, you should seriously consider the use of machine-readable turnaround documents. This will facilitate the recovery process in the event of a system failure, since a large amount of keypunching will not need to be redone. Re-keying already-entered data can be *very* destructive to the morale of an operator to whom computer input may be an unrewarding burden in any case (see User Training, below).

Using Machine-Readable Data

If you have chosen to use data capture to take advantage of existing machine-readable data, you will need to consider exactly how you will read the machine-readable information.

Bar coding Bar coding is becoming extremely popular in manufacturing as well as retail environments. It is easy and inexpensive to generate (and may in fact already exist, as in the case of the UPC/EAN codes in the retail field), easily read either under manual control or automatically by non-contact scanners, and is very reliable. Bar code readers are made by a large number of manufacturers including (in the United States) Interface Mechanisms (Intermec), Hewlett-Packard, Burr-Brown, Epic Data, and others. Some of these companies in addition manufcture complete data capture subsystems for inclusion on an existing computer system.

Bar codes may or may not be human-readable in addition to being machine-readable. Some bar code standards (UPC and EAN, for example) stipulate that the code shall have a human-readable translation above, under, or beside the code itself; you can, of course, design your system to provide the translation.

Optical Character Recognition Less popular than bar code is a system called Optical Character Recognition. Specially printed labels containing ordinary numerals (and sometimes letters as well) are read by a hand-held contact scanner. While the resulting label is smaller than a bar code label since the optical characters are typically more dense do not need translation, read reliability is rather poor with most systems, and several tries may be needed to read a label. OCR systems are usually manufactured as turnkey add-ons to existing computer systems. Some manufacturers of point-of-sale terminals offer OCR as an option.

Automated Test Equipment Automated test equipment systems (ATE) are becoming more popular as prices of versatile systems have dropped dramatically. By combining an ATE system with bar code or some other means of identifying assemblies, a completely automated testing *and* inspection records system may be set up. Interface to the HP3000 may be directly through RS-232C or RS-422 (terminal interfaces), or through GPIB or ordinary parallel interfaces using the HP3078 data collection terminal.

IV. Design and Implementation

System design aspects and their corresponding implementation aspects may be broken down into four areas:

o Topology
o User Interface
o Application System
o User Training

Topology

The network topology that you choose to implement depends on your needs as discussed earlier. The key questions are:

o Where is the data available?
o What form does it take?
o Who has the data?
o How will the data be communicated?

Where? On the principle that terminals should be located as close as possible to the source of the information that will be entered through them, you will need to locate as precisely as possible the locations in your plant which generate the information that your system is intended to capture. While this seems self-evident, all too often a location is denied a terminal because the usage would be low, even though the information generated there is as volatile as at higher volume locations. The result, of course, is "batching" of input and concomitant data timing problems.

Data capture terminals can usually be placed in crowded areas by using wall mounting. Hewlett-Packard's 3076A terminal is especially designed for permanent wall mounting; its keyboard and display may be used easily while sitting or standing. Wall mounting is also convenient when information will be entered by operators while walking or driving, as in access control or timekeeping applications.

What? The data to be entered can take many different forms. In many instances, data starts simply as knowledge: the receiving clerk knows that something has come in from a vendor, for example. In other cases, the data is part of an existing document, e.g., the packing slip contains a packing slip number that must be entered. If the data consists simply of knowledge, it is usually most expedient to simply key information into a convenient terminal. If data are contained on a preexisting document, it may be possible to code the document for machine reading. This is especially true if the preexisting document is generated within the company: the document can be generated on a printer capable of printing bar code to be read in later.

If the data to be captured exists in another computer (an ATE system, for example), no conversion or keying will be necessary.

Who? The data to be captured should be entered by the person (or object) most closely associated with it. In terms of the organization, the person or political entity that is *held responsible* for the accuracy, completeness, and timeliness of the tracked activity should have the responsibility for data entry. This means that terminals will be located close to the *people* handling the data: in the stockroom, on the shop floor, at the receiving dock, and so on. Where several people have knowledge of a particular physical event, the person who should be responsible for data entry should be the one who *must* have the knowledge for the business to run smoothly.

As an example, consider the shop floor controller moving assemblies on a shop floor. During normal operations, two people have the knowledge that an assembly has been moved to a new workstation. The shop floor controller, of course, is aware of it, but the because he or she caused the physical event. However, the assembly must be "formally" entered in the queue at the new workcenter. Thus, the workcenter's supervisor or lead operator, if any, should be responsible for data entry: the data capture system will then show the event only after action can be taken on it.

In some cases information will be subject to audit, and it may be desirable to limit update access. This may be accomplished with a machine-readable ID such as a magnetic striped badge or laminated bar code. In such cases, it may be expedient to use a method of physical identification similar to that which is used for automated data entry.

How? Once an event has taken place, is discerned to have done so by somebody, and a record of the event has been made and entered, the record must be transmitted to the computer. This is done either directly by the terminal, or by a data capture subsystem to which the terminal is connected. Hewlett-Packard, in general, favors the former approach; most other data capture vendors favor the latter. In practice, both may be used for different parts of the same data capture system.

In most installations, the data capture terminals will be located in the same building, or at least in close proximity to the computer acting as host. The discussions which follow assume that this is the case. If your situation is different, you will need to explore data communications options, but a full discussion of data communications is beyond the scope of this paper. The principles which will be discussed, however, are applicable to both remote and local data capture terminal networks.

For Hewlett-Packard data capture terminals, two options are available: multipoint connection, and point-to-point connection. Which of these options you choose should depend on:

- o Transaction volume
- o Volatility (i.e., minimum required reliability)
- o Proximity to the computer site
- o Cost
- o Flexibility

The first of these, and the one usually recommended by Hewlett-Packard, is multipoint. Hewlett-Packard offers two different hardware implementations of this software protocol.

Standard multipoint requires that all data capture terminals be connected in a "string," i.e., "daisy-chained" together. Cabling this arrangement over a large factory may be somewhat awkward, and the arrangement tends to be inflexible. Adding a terminal generally requires extensive additional cabling and in most cases will require that the data capture system be shut down during the installation. In addition, the method is limited in the distance that it can cover, and its immunity to electrical interference is somewhat limited.

The immunity of ordinary multipoint to single-point failure is limited by the length of the string. If a terminal fails, it will usually prevent operation of all terminals beyond it in the string.

The Factory Data Link, or multidrop, is an arrangement wherein many terminals share a common "supply" line, rather similar to houses along a water main. Physically, the data link consists of a fairly thick cable which may be initially installed as a loop around the inside of a building. The data link is limited to a maximum of five miles in length or less depending on the locations of terminals relative to the computer site. Data capture terminals are "tapped" into this line in much the same way that sprinklers are "tapped" into a garden hose. Each "tapped" terminal may have more terminals daisy-chained onto it as ordinary multipoint terminals. The method is very flexible, since installing a new terminal involves only tapping into the line at the point closest to the site of the new terminal, or daisy-chaining off an existing terminal if a second terminal is to be added to an existing data capture site. Adding a terminal disables only part of the system, and that only for a few minutes. Removing a terminal from the factory data link will usually not cause *any* downtime. Noise immunity characteristics of the factory data link are excellent; the data link will often function in environments with such high levels of radio frequency interference that no other connection method will permit error-free operation.

Immunity to terminal failure is very poor, however. Failure of a single data capture terminal can potentially "hang" all other terminals sharing the Factory Data Link.

Both multipoint connection methods, the daisy-chain and the Factory Data Link, are quite susceptible to single-point failure since many terminals share, at a minimum, a common computer interface and communication line. Failure of either of these components will cause the failure of all terminals in a network. For this reason, it is usually best to split a network into two or more "subnetworks" if a multipoint protocol is to be used.

If transaction volume is very high *and* the amounts of data transferred per transaction is large, performance can suffer. Because only one terminal can be using the common line at any given time, one very busy terminal can use most of the communication capability available. This must also be considered when designing the data capture network topology. But because of this facility sharing, the communication cost per terminal is quite low.

Point-to-point connection of data capture terminals has few of the disadvantages of multipoint connection. Immunity to terminal failure is excellent, since all common elements in the communication facility are thoroughly isolated from the terminal itself. Failure within the communication interface may easily be remedied by switching plugs *at the computer site*, an option not easily available with multipoint connection.

The chief disadvantages of point-to-point connection are the need to run separate cables for each terminal to be connected, resulting in decreased flexibility and significantly increased cost, and relatively low immunity to electrical noise.

Designing the User Interface

One of the most significant roadblocks in implementing a data capture system, and one of the least quantifiable, is simply *getting people to use the system*. One of the key considerations in this is the "user interface" - that part of the system which the user sees.

An almost universal characteristic of a data capture system is that the operators doing the most input usually have the least to gain from having the input done. This is in sharp contrast to batch systems or "on-line batch" systems in which the operators *primary* responsibility is to enter data. For the operator of the data capture system, then, maintaining the system is a burden rather than a part of the job.

By properly designing the user interface, this burden can be made as small as possible, and some element of reward can be introduced to make using the system more palatable. In general, the interface designer should be concerned with *simplicity* and *feedback*.

Simplicity Most data capture transactions will involve several steps, possibly implemented with a question-and-answer dialogue. If possible, this dialogue should be designed out of the user interface through the use of machine-readable documents. For example, consider the dialogue which might be required to move an assembly from one workstation to another. The important elements of the transaction might be the work order number on which the assembly is being built, the workcenter being moved from, the workcenter being moved to, and the quantity being moved. The system should generate all ancillary information such as the date and time of the move. In a dialogue, the transaction might be:

```
Computer: W/O#?
User:     4158245 <enter>
Computer: FROM OPN?
User:     45  <enter>
Computer: TO OPN?
User:     50  <enter>
Computer: QUANTITY?
User:     100 <enter>
```

This transaction requires 18 keystrokes and requires waiting for computer response after each of four entries. (These count a single initial keystroke for beginning the transaction.)

The transaction can be shortened somewhat by using defaults:

```
Computer: W/O#?
User:     4158245 <enter>
Computer: FROM OPN 45?
User:     <enter>
Computer: TO OPN 50?
User:     <enter>
Computer: QUANTITY 100?
User:     <enter>
```

This example requires only 12 keystrokes, since the application tries to anticipate the user's responses by checking existing information about the work order.

The transaction can be reduced still further by using a computer-generated work order form with bar codes pre-printed:

```
User:     (Wands composite FROM)
Computer: W/O# 4158245: 45 TO -?
User:     (Wands composite TO)
Computer: QUANTITY 100?
User:     <enter>
```

This example requires only one keystroke and two bar code reading operations. The "composite" is a bar code digit string containing the work order number, the operation number, and a transaction code meaning "move assembly." The operation code is ignored during input from the TO? query, e.g.:

041582450045m

Thus, by careful choice of user interface, 18 keystrokes and four waiting periods have been reduced to three keystrokes or user actions, and only two waiting periods. These waiting periods can be made quite short with proper application program design.

Feedback Because the primary responsibility of the data capture operator is typically something besides data entry, it is important that he or she receive some kind of feedback or reward in return for entering the data. This will usually consist of an acknowledging beep or message, but should include some additional information when possible. This may be difficult to achieve, but might come in the form of a count of units of production, e.g., number of units tested today or total from this lot. This serves to give the operator a sense that something inside the terminal *is* listening. Giving the operator a printed summary of the previous day's input can serve the same purpose. In general, the important point will be to insure that the operator realizes that some concern exists that the data entry part of the job gets done as well.

The amount of feedback should be proportional to the amount of work required for keying in. If the operator is simply required to run a bar code wand over a tag on the assembly, or to drop a card into a slot, little feedback will be required. If some dialogue is required, as in the above examples, more feedback should be provided.

Designing the Application System

Designing application programs for a data capture system requires attention to some aspects of programming that are not normally given much weight in ordinary batch or on-line processing. In particular, the programmer for a data capture system must pay attention to:

- High transaction volume
- High reliability and fail-soft operation
- High volume reporting

High transaction volume

When the number of transactions to be processed is very high, special attention must be given to such aspects of application design as file locking, transaction selection methodology, and general response time optimization. Because the key to all a successful data capture operation is a good user interface, and because a good user interface will derail the operator's primary activity for as short a time as possible, every effort must be made to shorten response time. An execllent paper on this subject was presented at the North American meeting of the HPIUG in 1983[1]. This paper covered various aspects of application design relative to response time and overall performance.

Using IMAGE

Of considerable importance to maintaining adequate transaction volume is the use of record-level locking when using IMAGE. Because many operators are likely to be accessing a fairly small group of high-use records, attention should be given to minimizing locking conflicts; locking at the lowest possible level is therefore a necessity. In addition, PUTs to detail data sets should involve primarily data sets with few paths. Sorted chains, unless used for maintaining chronological sequences, should be avoided. In addition, if several records on a chain must be manipulated, as in the case of the work-in-process tracking example above, the records should be manipulated in core to minimize the number of physical I/O's required to finish the transaction. Programmers should be willing to sacrifice simplicity for performance in this response-time critical situation.

Menu Selection

Many traditional menu selection schemes rely on process handling, creating and launching a new process for each transaction, or group of similar transactions. The overhead of this operation is unacceptable in for high transaction volumes and should be avoided. Other methods, such as using the menu (function selection keys on the terminal) to selectively call procedures in a single large program file, will improve response time and decrease I/O and memory resource usage. Since program code is shared among all users of a particular program, considerable simplicity of design can be achieved along with a performance increase by using this method. Common data base access routines should be placed in a single segment and shared among all transaction executors.

One helpful system design method is to avoid hard-coding menu parameters in your application program. If the program is designed to configure the function-to-key correspondence from a file, it becomes possible to provide each terminal in the data capture system with a subset of functions from a single large library. This feature becomes increasingly useful as your organization gains experience using the data capture system, since functions can easily be moved from terminal site to terminal site as the initial expectations inherent in the original design are modified.

Reliability

More than any other computer application, data capture systems require highly reliable software. While all software should be bug-free, data capture software with its limited communication back to its users, and its users' extremely limited computer training, is particularly dependent on non-stop, predictable, and accurate operation. The system should be designed to the greatest extent possible to be immune from problems from system failures, data base problems, program bugs, and especially user input.

Immunity from system failures can be provided in a number of ways. For a data capture system using IMAGE, the best way seems to be user logging. While requiring extra programming and extra design attention to be used effectively, transaction logging can provide up-to-the-minute backup for a busy data capture system.

Data base problems represent a broad class of difficulties such as broken chains in IMAGE data bases, DATA SET FULL errors and their equivalents in non-IMAGE systems, concurrency problems, and other difficulties in accessing or updating data on an otherwise healthy system. In general, the application should perform whatever checking is necessary to *commit the transaction to succeed.* This means checking data set capacities, insuring that records have not changed across locks, and perform whatever logical (data dependent) checks are necessary before actually updating the data base. Problems which are found should be reported to a central controlling process which can shut down the data capture system and inform the computer operator before any damage is done. In addition, the data capture system must *immediately* inform its operators that the system is no longer available. Otherwise, the operator may continue to "input" data, unaware that the system is no longer listening!

In order to prevent program bugs from causing serious damage, programs should be self-checking whenever possible. For example, if inventory counts are kept in several places (possibly split in several different ways), any time two or more counts are available, they should be compared. If an error is found, the problem should be reported to the operator, and depending on the potential seriousness of the problem, the data capture system should be shut down automatically. This strategy will prevent a "cancer" from occurring because of later transactions relying on the incorrect data.

In some cases, it is possible to repair damage to the data base in real time. For example, if it is possible to check a summary field against its details, this should be done whenever it is not inconsistent with performance. (This does not mean that the data check must execute rapidly; it may simply be performed as part of an infrequently-used function.) In the event of a discrepency, the damaged summary field should be repaired and the error logged. In this way, it is possible to insure maximum availability of the data capture system, while simultaneously maximizing its reliability.

High-volume reporting

In using a data capture system, a large amount of data can be recorded for later use. While it may seem obvious at the outset that these data will be too voluminous for repeated reporting, this fact should be kept in mind throughout the design of the data capture system. Reports should be provided with significant "filtering" capabilities so as to produce meaningful information from the huge amount of data collected.

User Training Program

Data capture systems present a special set of user training problems in that the system's users will typically have a lower level of computer education than a data entry operator. In order to mitigate this problem, the application should be designed so as to minimize the amount of training required. If a particular transaction requires more than one step to complete, the data capture system should guide the user through the transaction in such a way that only one path is possible. Error messages *must* be phrased in the user's "native" language; computer terminology should be avoided. *An uninterpretable error message will not be interpreted!* Failure to recognize this fact will mean lost transactions as users ignore rather than report cryptic error messages.

With this in mind, an effective user training program will be one which stresses training by example. For single-step transactions, the data entry becomes just one more step in a routine that the user is as already accustomed to as part of his or her primary duties. A multi-step transaction will require that the user understand something of the information that he or she is asked to present.

V. Evaluating your Data Capture System

After your system has been in operation for several months, you should begin to evaluate its performance and select changes to be made. Evaluating the system's performance relative to organizational objectives is beyond the scope of this paper, but four major operational areas should be evaluated:

o Reliability
o Operator acceptance
o Information availability
o User acceptance

Reliability

One repeatedly-stressed factor in a successful data capture system is its reliability. Uptime over the evaluation period should be measured, and any problems analyzed. Have there been communication failures? Have many problems been found by self-checking routines? Have the automatic repair and shutdown facilities been effective?

Operator acceptance

If the system has been reliable and available, it is important to measure operator resistance to the system. This is normally high at first, and then gradually decreases as data entry becomes a normal part of the operators' routine. (If the system has not been accessable, resistance will usually remain high regardless of other factors.) If resistance to use of the system is high after the system has been in place, the causes should again be analyzed. Have terminals been put in the right places? Are there enough terminals to prevent operators from standing in line? Is response time low enough? Is the user interface too complex, thus imposing a burden on the operator? Is insufficient feedback being provided?

Information availability

Since the primary purpose of a data capture system is to provide information which was too costly to gather with ordinary batch or semi-batch processing techniques, the first evaluation is whether the system is capturing the data it was designed to capture. If the operators have accepted the system, information should be available. If not, the design factors used relative to timely and useful presentation of information may need to be reexamined. Can reports be produced on an exception basis if desired by users? Can information on reports be tracked directly to physical activity monitored by the data capture system? Can information be obtained on-line so that users can "watch" the system work? Are users confident that they know the source of all of the data presented to them?

User acceptance

Finally, if information is available, properly presented, users should accept the system. Acceptance means *using* the results of the data capture system in their daily operations. If in fact decisions are made on the basis of information captured by your data capture system, it is by definition successful. After all, that is the end purpose of information systems!

Structural Engineering on the HP3000

- Creating an Environment for the Engineer

Authors

B Brennan
Technical Analyst
Information Technology Department
Britoil PLC
Aberdeen

G Stewart
Structual Engineer
Clyde Project
Britoil PLC
Glasgow

(formerly Information
Technology Department)

Abstract

Experience with engineering software at Britoil includes extensive use of mainframe packages but also a number of key systems are installed on the company's network of HP3000 computers. In particular many of the structural engineering calculations required by engineers within Britoil may be handled on the HP3000.

To provide an effective environment for stress calculations requires careful attention to the way information is handled. The Britoil Stress System comprises a set of inter-linked programs and packages several of which require the maximum stack allocation.

Significant usage is made of the structural graphics packages FEMGEN and FEMVIEW to generate data and view graphical output. These are linked to the main structural solver STRESS with pre- and post- processors which include bandwidth optimisation and API code checking. To provide a fully integrated system which engineers can use without having a detailed knowledge of the computer system presented an interesting challenge for engineering software specialists.

Particular problems were encountered:-

-Linkage of 10 systems using the restricted 32K stack size

-Emulation of a TEKTRONIX terminal programmatically

-Interaction with an HP7220 series plotter when the terminal is used in TEKTRONIX emulation mode.

By extensive use of MPE Intrinsics an environment is created for the structural engineer which minimises his problems of file management and considerably decreases the amount of information he is required to know about the computer system itself. The resulting system is independent of terminal type and requires no special knowledge of particular configurations of printers and plotters found in different locations. Each individual user manages his own set of files used for input to the packages and for communication between related packages. Access to the MPE System Editor is provided within the system to allow manipulation of input data files. The system is now being used by structural engineers throughout the Corporation.

Acknowledgements

The authors wish to thank Britol PLC for permission to publish this paper. They also express thanks to those engineers within the company who have contributed to the specification and success of the work.

Background

As the activities of the British National Oil Corporation developed in the late 70's, it became apparent that structural engineering software was required. Initially this requirement was met by accessing Bureaux packages but due to the level of usage and the resulting support required it became desirable to provide an in-house facility.

The structural analysis program STRESS was acquired from the HP USER CONTRIBUTED LIBRARY and mounted on two of the Corporation's then three HP3000 Series 3 computers (at this time the Corporation had no mainframe). This program proved to be reliable and was soon being used extensively throughout the Corporation.

Users of structural analysis programs encounter two main problems:

- setting up the complex input data required and ensuring its integrity
- analysing the mass of output produced to identify the areas of particular interest

After much consideration it was decided that the software package FEGS* (Finite Element Graphics System) be acquired to relieve these two bottlenecks. The system was duly purchased and mounted on the relevant HP3000's (early 1981).

With the aid of the in-house written pre and post processors the STRESS program became the nucleus of a complete structural analysis system and the FEGS package the means through which the engineers could graphically generate their data and interrogate the results of analysis.

Fig.1. shows the typical order in which STRESS, FEGS and the several in-house packages would be used to carry out a complete structural analysis. At that time (early 1982), each of these packages was stand-alone with its own command procedure and file allocations. The consequent multiplicity of commands and files required led to problems in usage of the complete system. In addition use of the graphics and plotting facilities required that the user have different specific terminal types for each operation which proved extremly clumsy in practice and needed improvement.

These engineering functions of the Corporation were part of the activities transferred to the new company Britoil during 1982 and it was after this date that proposals to upgrade the STRESS system were implemented.

* FEGS Limited,
2 All Saints Passage
Cambridge
CB2 3LS

Description of System

The Britoil STRESS system comprises of a set of inter-linked programs and packages several of which require the maximum stack allocation of 32K. Significant usage is made of the structural graphics packages FEMGEN and FEMVIEW to generate data and view graphical output. These are linked to the main structural solver STRESS with pre and post processors which include CPU optimisation and API & AISC checking of resulting stresses.

Computer Museum

Fig.2. shows an outline of the Britoil STRESS system:
- it enables the user to run through a complete structural analysis easily, minimising his problems of file management and considerably decreasing the amount of information he is required to know about the computer system itself

Initially a user is presented with the Main Menu (Fig.3.), Choosing option 1, an analysis program, leads to a further menu (Fig.4.). After selecting a specific program from this menu he is prompted for required file names. The main program is then suspended during operation of the analysis program and upon its completion re-assumes control.

Option 2 (HELP) causes display of a specific Help Menu (Fig.5.). Choice of an option from this menu will cause a brief description of that program's function/s to be displayed. This Help information (Fig.6.), is stored in a central file to facilitate changes and extensions to the system.

Selecting the system OUTPUT option (3) displays the OUTPUT menu which offers the choice of terminal, local matrix printer or the printer output.

Option 4 (Plotting), enables the user to obtain a hard-copy by a local flat-bed plotter of graphical displays previously set up within the FEGS suite (JARGEN, FEMVIEW).

Choice of Option 5 (Editor), provides access to the HP3000 EDIT facility with its full range of capabilities. Upon termination of the EDITOR,control is returned to the Main Menu.

Option 6 (Amendments to Previous Version), accesses the central file, retrieves a list of differences/enhancements between the current and previous versions of the system, and displays the AMENDMENTS information on the screen. The current version number is displayed at the top of the Main Menu.

The engineering users concerned have access to a varied hardware environment including the following:-

HP2623	Graphics terminal
HP2626	Terminal
HP2645	Terminal
HP2647	Graphics terminal
Tektronix 4014	Terminal with linked thermal hardcopy
HP7220	Flatbed plotter
HP9872	Flatbed plotter
HP7580B	Drafting Plotter
HP2631	Local Printer

The Stress System is available on an HP3000 Series 64 in Glasgow and an HP3000 Series 33 in Aberdeen. These machines are part of Britoil's DCX Network which allows communication between the different Britoil staff locations and the computer facilities spread amongst locations.

Engineers use the system on their local machine but in the event of this being unavailable they have the opportunity of using an alternative machine.

The following facilities are built into the system

- Switch Terminal Emulation where required by graphics/ plotting software
- Print files locally or remotely
- Edit data files
- Advice to user of completion of Stream Jobs

These facilities provide the engineer with his own operating environment without the need to understand systems intrinsics.

Software Capabilities

Objectives

The capabilities required to link the BRITOIL STRESS SYSTEM together are:

- running programs within programs
- alter Terminal Emulation from within a program
- create, allocate, purge and stream files from within a program; also to check their status
- alter Terminal Characteristics (eg ECHO)
- advise of completion of relevant stream job
- the more changeable aspects of the system should not necessitate recompilation if modified.

Solutions

The infrastructure of the BRITOIL STRESS SYSTEM is linked together by a FORTRAN program (~2000 lines) which controls file allocation, terminal characteristics etc and becomes the Users Operating System.

The methods used to provide the facilities required are:-

MPE Intrinsics CREATE and ACTIVATE - Used to suspend main program and run existing stand-alone software (STRESS, FEGS, FCOPY, EDITOR etc.)

MPE Intrinsics FNUM, FCONTROL - Used to interrogate terminal type and alter if required for graphics.plotting (FEGS/CALCOMP)

MPE Intrinsics FOPEN, COMMAND, FCLOSE, WHO - Used to Create, Allocate, Purge and Stream files; also to check their status.

FCONTROL + Terminal Display Sequences - Used to alter terminal characteristics and send messages to the graphics screen

Central File Contents

All sections of the system of a more dynamic nature are held in a central file rather than the source code and picked up during program execution
These include:

- Analysis Program names
- Several Display statements eg Main Headings
- Help information
- Amendment Information
- Error messages and their level of fatality

This avoids continual recompilation of the source for minor changes and gives the Systems Controller the ability to alter program internals during execution.

Example Of Use

Consider the following example of the system in use.

The simple structure shown in fig 7 is generated and analysed as follows:-

a. Enter the Analysis Program Menu via option 1 of the Main Menu, then select FRAME (option 1).

   This interactive program generates a COMMAND FILE definition of the model for subsequent use in JAKGEN.

   The user supplies the minimum of information necessary to define the structural link points (nodes) as shown in Fig 8 and FRAME automatically generates the connecting members. (fig 9).

b. Select the JAKGEN option to switch the terminal to graphics mode. Having read the command file into JAKGEN the model can be altered using the cursor - for example in Fig 10 the diagonal members connecting the nodes 3-6, 4-5, 0-12 have been added.

c. When satisfied that the model geometry is correct, proceed to the option PRESTRESS, an interactive program, to define the physical aspects of the model such as member diameters, applied loadings and boundary conditions.

   This program will automatically generate the self weight of the structure and give the coordinates of the centre of gravity.

   This program generates the data file for input to the structural solver STRESS.

d. The user may wish to inspect the STRESS input data file using the EDITOR before continuing. Once in EDITOR the model definition can be changed manually.

e. Having defined the model and its attributes submit job via the STRESS option and await output on disc file.

   The output files can be printed automatically and post processing of results is also an option. This is shown in Fig 11

f. When the JOB has terminated the user may wish to interrogate sections of the output graphically. To achieve this the output file is first converted into a standard ASCII format using the POSTSTR option. This also causes the program PREVIEW to run which transfers the ASCII file into the FEMVIEW DATABASE.

g. Select FEMVIEW to interrogate the output using a menu. For example figs 12 and 13 show the displacements and the axial forces from the analysis.

   Using the zoom facility (fig 14) areas of complicated geometry can be magnified until they become clear.

   Thus the user can rapidly decide if the structure has responded in an acceptable way and identify any areas which cause concern.

Conclusions

Several further developments of the system are likely, amongst these are:

- Link to a Mainframe Finite Element package (either via MRJE or IMF)
- Use of the system building blocks in other engineering areas, eg Pipe Stressing, Process.

The system has proven its usefulness by significantly decreasing turnaround time for engineering whilst:

- considerably increasing the engineer's confidence in the system and giving satisfaction with its operation
- decreasing the level of support required for the system by a factor of 2
- reducing the amount of training required for new users to the extent that a SYSTEM USER MANUAL has proven to be unnecessary.

Note

For further information concerning this paper contact.

Dr N Rawlings
Engineering Systems Section Controller,
Information Technology Department
Britoil PLC
150 St Vincent Street
GLASGOW G2 5LJ

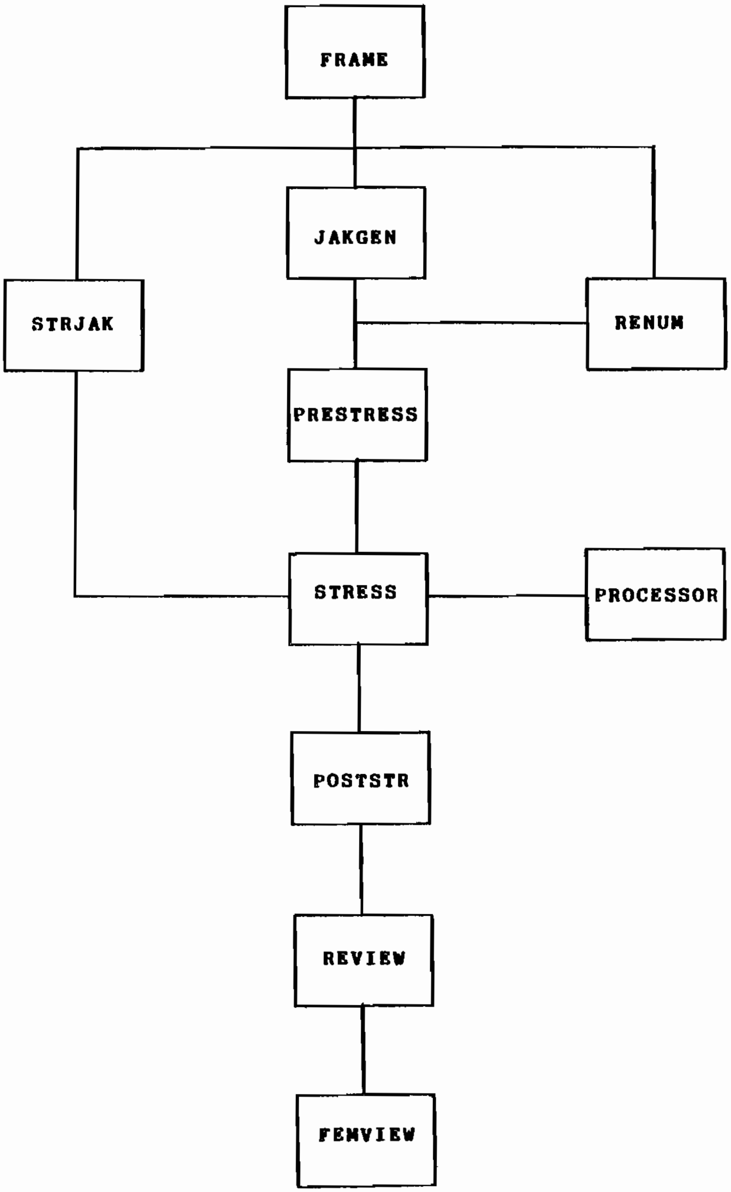

FRAME
JAKGEN
STRJAK
RENUM
PRESTRESS
STRESS
PROCESSOR
POSTSTR
REVIEW
FEMVIEW

Fig 1 - Logical Order Of Use In Full Analysis

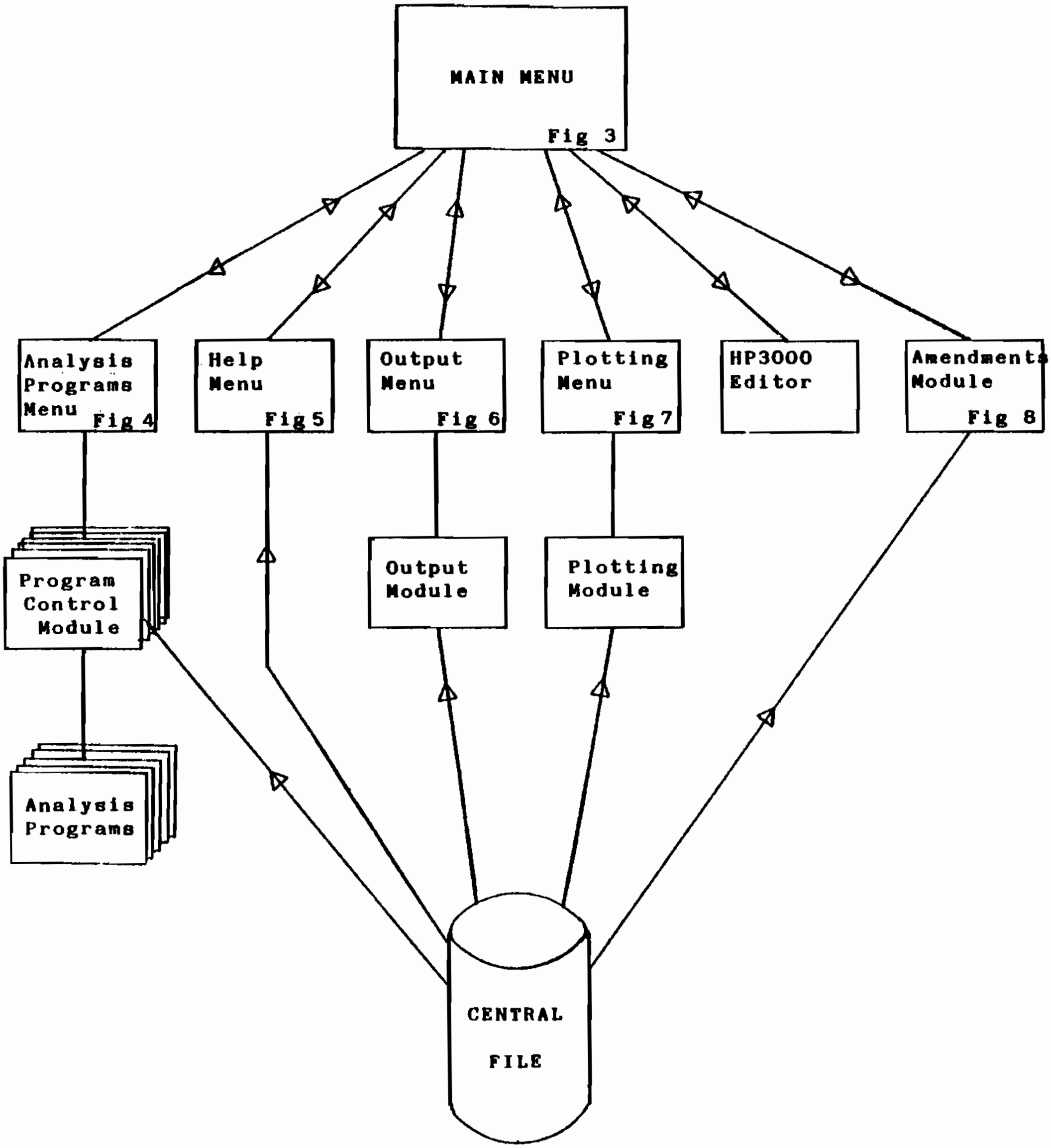
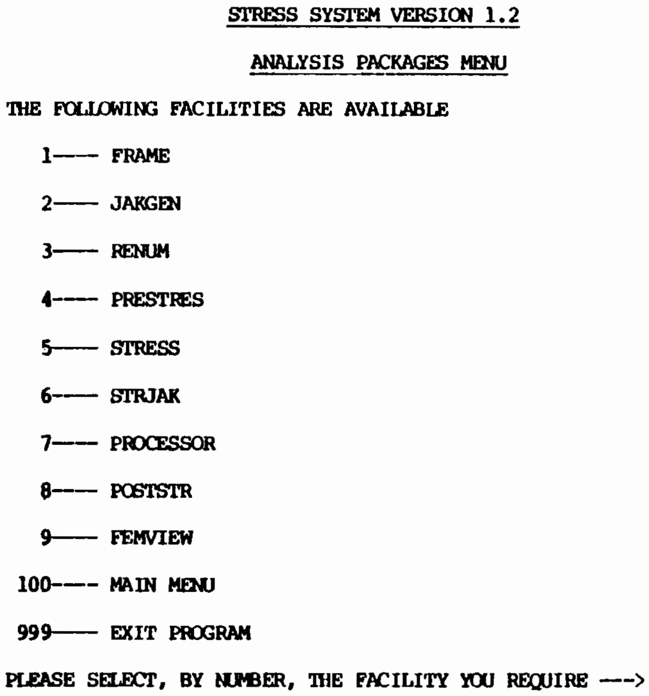

MAIN MENU
Fig 3
Analysis Programs Menu
Fig 4
Help Menu
Fig 5
Output Menu
Fig 6
Plotting Menu
Fig 7
HP3000 Editor
Amendments Module
Fig 8
Program Control Module
Analysis Programs
Output Module
Plotting Module
CENTRAL FILE

Fig 2 - Outline Of Britoil Stress System

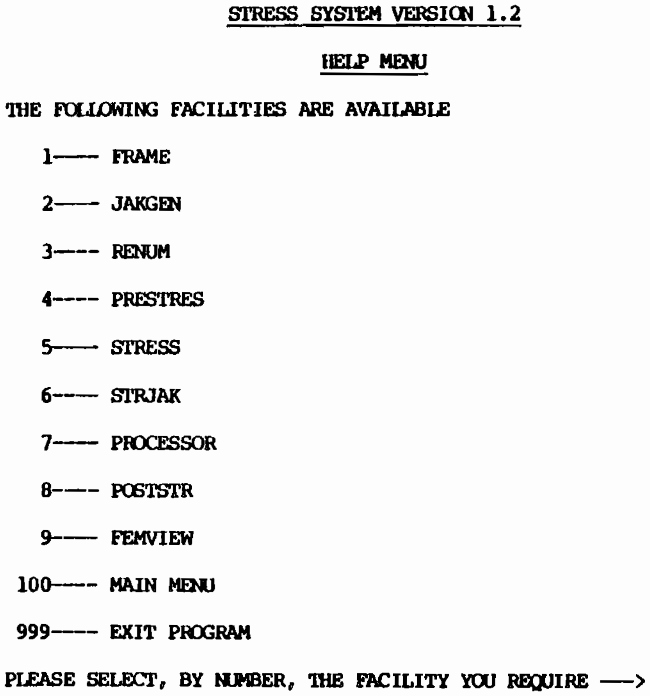
```
                    STRESS SYSTEM VERSION 1.2

                            MAIN MENU

THE FOLLOWING FACILITIES ARE AVAILABLE

  1 ---- ANALYSIS PROGRAMS (STRESS ETC.)

  2 ---- HELP

  3 ---- OUTPUT

  4 ---- PLOTTING

  5 ---- EDITOR

  6 ---- AMENDMENTS TO PREVIOUS VERSION

999 ---- EXIT

PLEASE SELECT, BY NUMBER, THE OPTION YOU REQUIRE --->
```

Fig 3



```
                    STRESS SYSTEM VERSION 1.2

                     ANALYSIS PACKAGES MENU

THE FOLLOWING FACILITIES ARE AVAILABLE

   1---- FRAME

   2---- JAKGEN

   3---- RENUM

   4---- PRESTRES

   5---- STRESS

   6---- STRJAK

   7---- PROCESSOR

   8---- POSTSTR

   9---- FEMVIEW

 100---- MAIN MENU

 999---- EXIT PROGRAM

PLEASE SELECT, BY NUMBER, THE FACILITY YOU REQUIRE --->
```

Fig.4.



```
                         STRESS SYSTEM VERSION 1.2

                                HELP MENU

THE FOLLOWING FACILITIES ARE AVAILABLE

   1---- FRAME

   2---- JAKGEN

   3---- RENUM

   4---- PRESTRES

   5---- STRESS

   6---- STRJAK

   7---- PROCESSOR

   8---- POSTSTR

   9---- FEMVIEW

 100---- MAIN MENU

 999---- EXIT PROGRAM

PLEASE SELECT, BY NUMBER, THE FACILITY YOU REQUIRE --->
```

Fig 5

- FRAME is used to set up frame type structures before entering JAKGEN.

- The user defines the structure by specifying the number of bays in the X, Y and Z directions.

- JAKGEN is an interactive data generator. The user defines his structure using points, lines, nodes and elements which may be partitioned into SETS. The MESH command is used to define the ELEMENTS and NODES.

- RENUM is used to obtain the optimum bandwidth for a model previously created by JAKGEN.

- PRESTRFS accesses a file containing the JAKGEN MODEL definition. It is an interactive program which sets up a STRESS input file.

- STRESS is a structural analysis program for the analysis of BEAM TYPE structures.

- STRJAK converts the geometry of the STRESS input file which defines the model to JAKGEN. This file can be read into JAKGEN using the PROC READ 10 command.

- The PROCESSOR program performs CODE CHECKING to API and RP2A for tubes and I beams. The program reads the STRESSS input and output files.

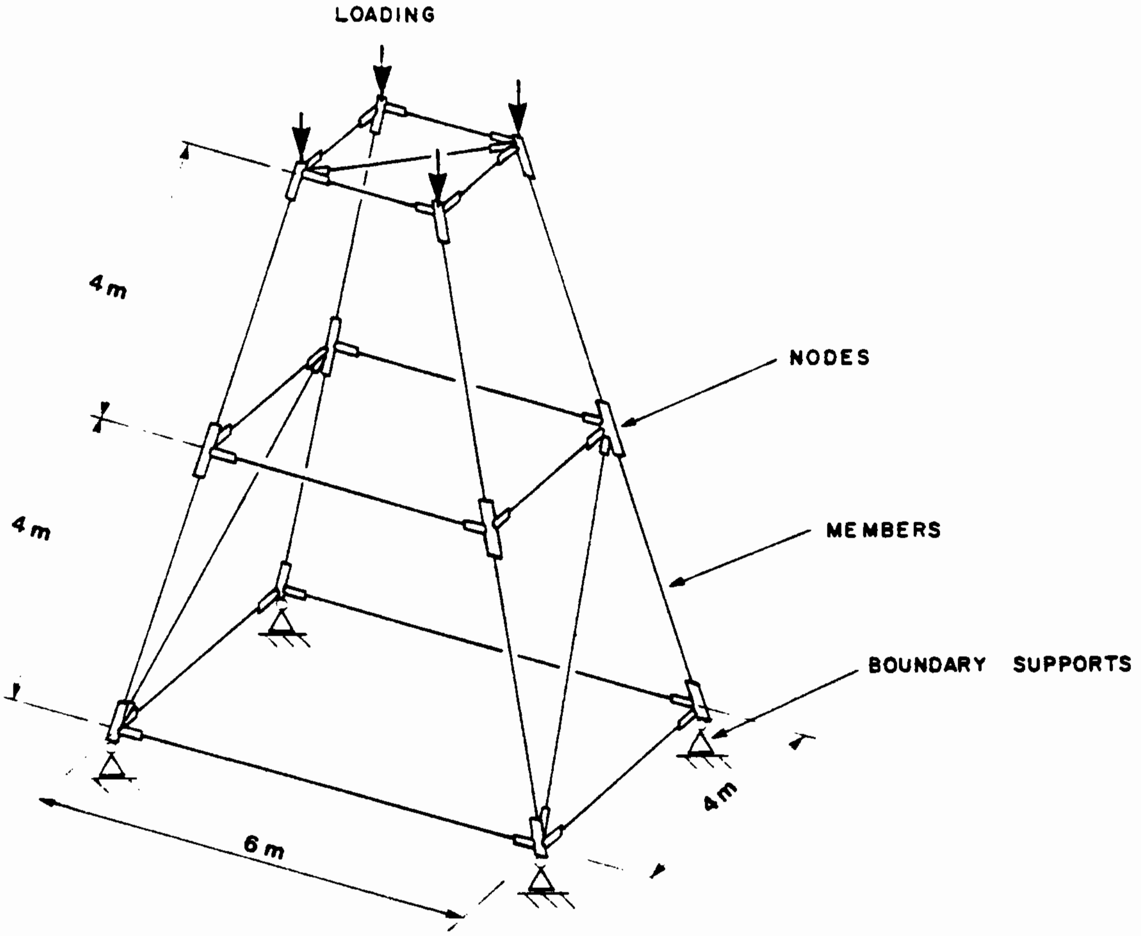
- POSTSTR is a conversion program to put data into a form suitable for FEMVIEW. (Selecting this option also causes execution of PREVIEW).

- FEMVIEW is a results viewing package, giving output plots of DISPLACED SHAPE, AXIAL FORCES, REACTIONS etc. A USER NAME is required to enter the program.

- Contact IT to obtain one.

Fig 6 Example of Help Information

Fig. 7


LOADING
NODES
MEMBERS
BOUNDARY SUPPORTS
4 m
4 m
6 m
4 m

```
ENTER NAME OF FRAME OUTPUT FILE - NEW —> HPFR1

ENTER NUMBER OF FRAMES IN X, Y AND Z DIRECTIONS
(ENTER 1 FOR NUMBER IN Z DIRECTION IF PLANE FRAME REQURIED)

        NX     NY     NZ
—>       2      2      3

ENTER X, Y COORDS.  OF THE 4 BASE CORNERS

     (X1, Y1)   (X2,Y2)   (X3,Y3)    (X4,Y4)
-->  -3   2       3  2    -3  2       -3  -2

ENTER X, Y COORDS.  OF 4 TOP CORNERS

     (X1,Y1)    (X2,Y2)   (X3,Y3)   (X4,Y4)
—>   -1  1        1  1      1  -1     -1 -1

ENTER HEIGHT OF EACH DECK (INCLUDE THE BASE)

     H1   H2   H3
—>   0.0  4.0  8.0

TO CONTINUE - - PRESS RETURN
```

Fig 8 - Model Definition Using Frame

```
→ PLOT G ALL 60 30

→

→ PLOT P
```

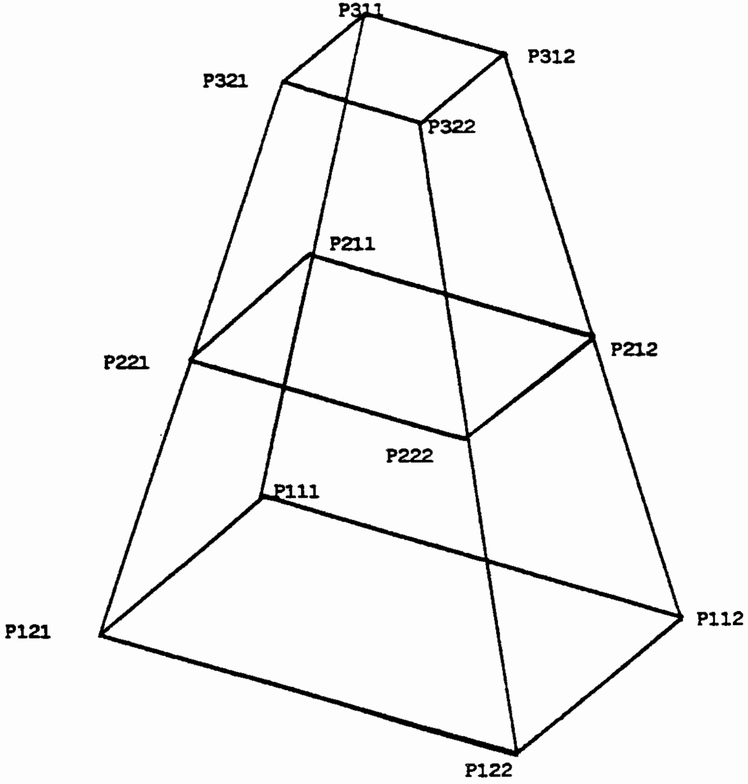

P311
P312
P321
P322
P211
P212
P221
P222
P111
P112
P121
P122

FIG 9 MODEL GENERATED USING FRAME (PLOTTED IN JAKGEN)

```
→PLOT  M

→PLOT  N
→
```

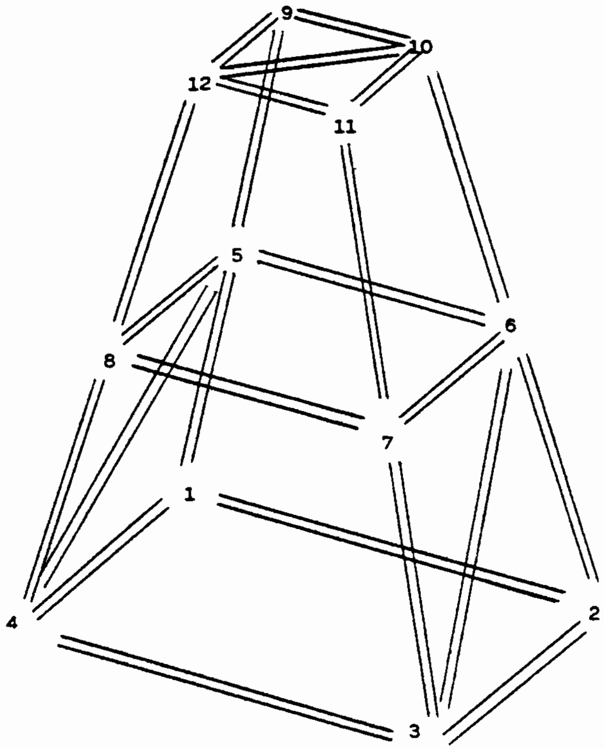

9
10
12
11
5
6
8
7
1
4
2
3

FIG 10 MODIFYING THE MODEL WITHIN JAKGEN

```
ENTER YOUR USER PASSWORD, IF ANY --> STRUCT

ENTER YOUR ACCOUNT PASSWORD, IF ANY -->

ENTER NAME OF STRESS INPUT FILE - EXISTING --> HPSTRIN

ENTER NAME OF STRESS OUTPUT FILE - NEW --> HPSTROUT

DO YOU WISH YOUR OUTPUT FILES PRINTED AUTOMATICALLY? (Y/N) --> N

DO YOU WANT CODE CHECKS? (Y/N)
  (STRESS POST PROCESSOR) --> N

J381
```

<u>Fig 11 Submission of Stress Job</u>

CAPITAL PROJECT CONTROL SYSTEMS

PRESENTED BY DAVID HARVEY
BRITOIL PLC

David Harvey is the Section Controller of the Capital Project Systems Development Group responsible for the development, implementation and support of Commercial Systems used during the Design, Construction, Hook-Up and Commissioning Phases of Britoil Operated Projects in the North Sea.

CONTENTS

1. INTRODUCTION

2. SCOPE OF THE PROBLEM

3. CONCEPT OF INTEGRATED SYSTEMS

4. A TYPICAL CAPITAL PROJECT SYSTEM

5. SYSTEMS IN A TYPICAL PROJECT

6. PRACTICAL PROBLEMS

7. POTENTIAL BENEFITS

1. INTRODUCTION

Britoil is a major new British oil company formed from the oil and gas interests of the British National Oil Corporation.

Britoil's business is in the exploration, development and production of oil and gas.

With the vast amount of money which needs to be invested in the development of any major field in the North Sea, such developments are normally undertaken by groups of oil companies acting in partnership. Each Partner contributes a share of the field development costs and in return obtains a corresponding entitlement to the production of the field.

One of the Partners is nominated to be the Operator for the development and undertakes the task of developing the field on behalf of the other Partners in the consortium.

Britoil's main assets are its interests in six producing oil fields, in one producing gas field and in three oil fields under development.

Britoil is Operator on 4 major developments :

- Thistle	(in production)
- Beatrice	(in production)
- North Leg Gas Pipeline	(commissioning phase)
- Clyde	(design phase)

The Capital Projects Systems Group was formed in 1978 to embark on the development of an integrated set of Project Control Systems to support these and future Britoil Operated Capital Projects through the development phase of the project covering the design, construction, installation, hook-up and commissioning to the stage where oil and gas are being produced and the facility is handed over to the Operations Group.

2. SCOPE OF THE PROBLEM

An idea of the volume of data generated by a typical large North Sea Project can be derived from the statistics which were compiled from the Thistle Project :

- 750,000 planning activities
- 20,000 design drawings
- 700 major design revisions
- 25,000 equipment and material specifications
- 10,000 steel plates
- 27,000 purchase orders
- 200,000 material movements
- 250 major contracts
- 150,000 invoices

The problem was compounded by the geography of a typical North Sea Project. On Thistle there were five major design offices and ten major construction sites both in the UK and on the Continent. Information generated in any of these locations could be required in any of the others.

3. CONCEPT OF INTEGRATED SYSTEMS

The objectives of Computer Systems Support to a Capital Project is to provide information of sufficient quality and in a suitable timescale to enable Project Managers to take effective decisions. These decisions assist in the achievement of Project Management objectives

- to complete the development on time
- to complete the development within budget
- to ensure a smooth transition to the Production Phase

One of the major problems which has hindered the decision making process has been the fact that information, although available, has seldom been in a form which would allow meaningful comparisons to be made and variances to be identified.

Stand-alone computer systems, although of benefit within a particular functional area of a Project, were seldom capable of providing information of significant value to users in other functional departments or to Project Management.

The concept of the Integrated Project Control System within Britoil was of a set of stand-alone modules which would fully meet the needs of particular management areas and which would enable information to be provided to a central Project Control function.

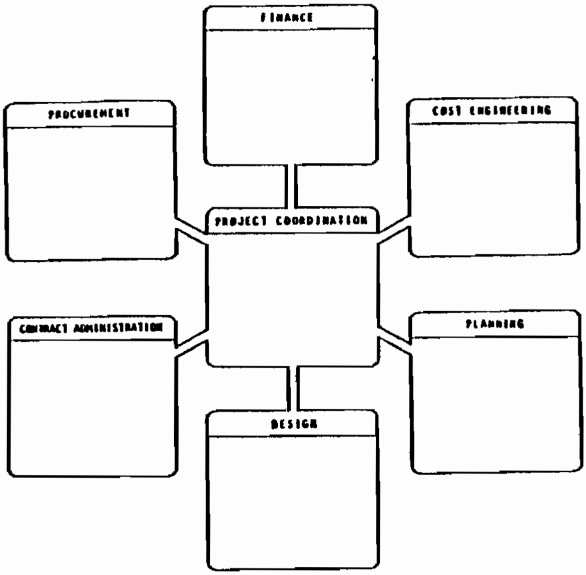
The Capital Projects Group was therefore set up to provide a set of computer systems within the functional areas of :

- engineering
- planning
- cost engineering
- materials
- contracts
- finance

which were capable of providing information identified in a way which would enable :

- budget
- commitment
- cost incurred
- expenditure
- estimate to complete

to be reported in a consistent manner, thus allowing variances between them to be identified and enabling corrective action to be taken.



The major innovation within Britoil, which enabled the process of integration to become a possibility, was the adoption of a common coding structure throughout the Project. This code, known as the Project Assignment Code, allows the physical structure being built to become the basis for budgetting, for the measurement of progress and for the collection of costs. The Project Assignment Code has become the vehicle for the integrating of management information and for the reporting to different levels of Britoil and Partner management.

The other major decision which enabled the development of an integrated set of computer systems to become a reality was the decision to standardise on the Hewlett Packard HP3000 as the Project 'mainframe' for commercial applications and the introduction of a network of communications which would allow terminals in any of the Project locations to communicate with the various application systems on whatever computer these applications happen to have been developed.

Some of the engineering software packages, for example, are mounted on the Britoil IBM3033 mainframe. The Planning package in current use within Britoil is the Metier Artemis System mounted on dedicated Hewlett Packard HP1000 mini-computers.

From a single terminal in any of the Project offices, it is possible using the communications network to access any of these applications and to initiate the transfer of information between applications mounted on different machines.

(This communications network is the subject of a separate Britoil presentation to the IUG).

The bulk of the systems development which has taken place has been carried out by Britoil's own systems development staff.

On the HP3000 the major systems development tools which have been used have included :

- COBOL
- VIEW
- IMAGE
- QUIZ (Quasar Systems, Ltd)
- PROTOS (Cole and Van Sickle Company, Inc)
- DI3000 (Precision Visuals, Inc)

On the IBM3033, the development tools have included :

- FOCUS (Information Builders, Inc)

4. A TYPICAL CAPITAL PROJECT SYSTEM - CEMS

One of the first systems developments which took place within the Capital Projects strategy was the development of a material control system. Material traditionally accounts for 15% of the cost of a North Sea Project but the knock on effect of late delivery of material can be out of all proportion to the cost of the material involved.

The development of the system, CEMS (Control of Equipment and Materials) again depended upon the adoption of a materials coding system which would allow material requirements to be compared at item level with the material availability from stock (including surplusses from previous Britoil projects and operational spares procured by the Operations Division), from purchase order and from requisition.

For this, Britoil adopted the Shell MESC (Materials Equipment and Standards Coding) System, already in common use throughout the oil industry, as the basis of its part number identification.

The CEMS System although primarily developed as a material control system contained sufficient information to enable it to provide the value of commitments, the value of cost incurred for materials and the basis of short term future costs derived from anticipated deliveries. To ensure that it could do this in a meaningful way, it is necessary that all materials information is identifiable to the Project Assignment Code and that estimates of costs prior to purchase order placement and actual purchase order values are recorded.

Whereas this information may be of limited value to the day-to-day procurement, expediting and movement of material, it is of vital importance in the provision of management information. It does however require a significant level of commitment and discipline within the materials area to ensure that this data on which management information is provided is of the requisite quality.

The CEMS System itself contains information about all Project Material from the stage at which an engineering requisition is raised, through the bid evaluation cycle, through the approved authority to purchase, through the purchase order, expediting traffic and quality assurance stages culminating in the delivery and receipt of material at the construction site. The issue of material to a fabrication contractor completes the material control cycle.

This information identified by MESC and assignment code is available for comparison with the material requirements similarly identified and thus enables material shortages to be identified, hopefully in a timescale which will enable the adverse effect of these problems to be eliminated.

The specification for the system was prepared in conjunction with the Corporate Materials function on the basis of experiences during the Thistle Project. At the time of this specification, there was no current Britoil Operated Capital Project in existence. The specification was based on the assumption that Britoil would be responsible for the procurement of all material on future Operated Projects.

5. SYSTEMS IN A TYPICAL PROJECT

The North Leg Gas Pipeline Project was the first of the Britoil Operated Projects on which the integrated systems approach was attempted. The Project is for the construction of a pipeline system to collect gas from the Murchison, Thistle and Magnus fields and to transport it to the FLAGS pipeline before onward transmission to its final destination at St. Fergus.

The Systems which were developed in the various functional areas were :

- the CEMS materials system described earlier.
- the Company system for the preparation and presentation of Budgetary information. The system was modified to allow the previous cost code field to hold the Project Assignment Code and to provide information identified to this code.
- a planning system based on Artemis. At the budget preparation stage , this was used to prepare the Project Plan and to provide manhour estimates for inclusion in the Company budget preparation system. At the regular weekly/monthly progress progress cycles the system provided information which enabled the Project estimate to complete and anticipated final costs to be calculated.
- a simple contracts management system providing commitment and cost incurred information identified to the Project Assignment Code.
- the Company Accounts Payable system which provided details of paid/unpaid costs identified by assignment code and contract/purchase order number.
- the Company Financial System which provides details of payroll costs, employee expenses and overheads. This information had to be provided, identified in a way that it is compatible with the Project Assignment Code.

At the end of each month, information from each of these systems was transferred into the Integrated Project Control System. After correction of obvious errors which had been introduced because of incorrect application of assignment codes, the monthly set of management reports were provided for :

- NLGP Management
- Britoil Execution
- NLGP Partners

Examples of the type and format of report are :

- Cost Status Report showing relationship between Budget, Approved AFE's commitment, expenditure estimate to complete and anticipated final cost at assignment code level.
- Purchase Order/Contract Status Report showing commitment and cost incurred information by purchase order/contract.
- Graphs showing actual commitment against planned commitment for both contracts and purchase orders.
- Graphs showing actual expenditure against planned expenditure for both contracts and purchase orders.

6. PRACTICAL PROBLEMS

- Integration is a valid objective. It is often more difficult to achieve than anticipated. It requires a high level of commitment from a number of different functional areas who may not derive direct benefits from the additional work required or additional information which needs to be maintained within their functional system.

- Assumptions, on which the development of integrated systems takes place, have to be confirmed at the highest level. The systems have to be flexible to allow for the possibility that these assumptions may change. In our case, the materials systems were built on the assumption that Britoil would procure all Project materials with only minor exceptions. This has happened on the previous Britoil Projects, Beatrice and North Leg Gas. There are sound commercial reasons why the Design Contractors on the Clyde Project will procure most of the material and the systems have been modified to reflect this decision.

- Interfaces between the various functional systems and with the Integrated Project Control System have to be clean so that system problem in one functional system does not impact on the ability to provide integrated management information. It also allows for the possibility that on a future Project it may be a requirement for a contractor's own system to provide information for inclusion in the Integrated Project Control System.

- Validation of information across traditional system boundaries can often prove to be less than satisfactory. The speed with which events can happen sometimes, thankfully not frequently, mean that transactions cannot be registered in one application system because for whatever reason, the prerequisite transactions have not taken place in other functional systems. This can be a useful discipline and check that systems are being regularly updated. It is imperative that it does not become the source of friction between functional groups on a Project.

- The introduction of Coding Systems within an established organisation can be an extremely demanding process. There is never a right time to introduce such a system, but the overiding consideration has to be that if a coding system is to be introduced it should be introduced at the earliest possible time after the need and the format of the code have been agreed. The extent to which previous information should be coded to the new standard should be determined by the potential advantages to be derived.

- It is important that the coding systems which are developed should have realistic objectives. There is no point in having a 15 digit code if management information is based on and control exercised at the first 3 digit level. The codes have to be useable by the people who have to apply them. Their use occasionally has to be reviewed in the light of experience.

- Systems have to be developed in such a way that recovery from hardware failure, telecommunications failure, or software error in a functional system can be accomplished with minimal disruption to other functional systems. As reliability improves, user expectations of service levels can become unrealistic, and when failures almost inevitably occur, the work of the user departments can come to a grinding halt. It is imperative that in the case of failure, further access to possibly corrupt or inconsistent databases is not permitted until the recovery is complete. The more integrated the processes, the more difficult recovery can become.

7. BENEFITS

The potential benefits which can stem from the adoption of an integrated set of Project Control Systems include :

- management information can be provided as a natural by-product of functional systems where sensible coding systems can be adopted on a Project wide basis
- management control information, budgets, commitments, cost incurred and progress information can be presented in a consistent manner
- areas of potential problem can be readily identified and corrective action initiated in a timescale which is likely to be effective
- management decisions can be made within the framework of accurate and up-to-date information
- consistent information is reported at all levels within a Project and to Britoil Executive and Partner management
- unnecessary duplication of information, and the duplication of effort in compiling information can be avoided

Britoil announced on 19th July 1983 that the North Leg Gas Pipeline was commissioned and that first gas had reached the FLAGS System, three months ahead of schedule and within Budget. Following successful commissioning, the pipeline will be handed over to British Petroleum, the Operator for the production phase. NLGP Management have been enthusiastic supporters of the concept of the Integrated Project Control System. They have placed on record their belief that the use of such systems has made a contribution to the success of the Project.

The systems on the North Leg Gas Pipeline Project were an early step in the evolution of Capital Project Systems within Britoil.

APPENDIX - OVERHEAD PROJECTOR SLIDES

1. BRITOIL NORTH SEA INTERESTS.
2. STATISTICS OF A NORTH SEA PROJECT.
3. GEOGRAPHY OF A NORTH SEA PROJECT.
4. INTEGRATED SYSTEMS CONCEPT.
5. CLYDE PROJECT ASSIGNMENT CODE.
6. BRITOIL DCX COMMUNICATIONS NETWORK.
7. BRITOIL COMPUTER HARDWARE.
8. CEMS SYSTEM OVERVIEW.
9. NLGP PROJECT SYSTEMS OVERVIEW.
10. NLGP SAMPLE REPORTS.
11. PRACTICAL PROBLEMS.
12. BENEFITS.

A Case Study on the Installation of Manufacturing Systems

(Bonas Machine Company Ltd, 1978 to 1983)

R J Woodhead, B.Sc. MBPICS

1.0 INTRODUCTION

This case study is concerned with the installation of a HP3000 computer system at Bonas Machine Company and the manufacturing software necessary to solve the company's stock control and material planning problems. For cost effectiveness an application package was chosen. Initially, this was the HP MFG/3000 system which was subsequently upgraded to Materials Management/3000.

The reason for documenting a case study is not just for the historical record, but so that others may learn from the successes and the failures. This paper is intended for those HP3000 users who are about to implement an application package such as MM/3000. It is important to compare theory with practice and to review with hindsight, the major policy decisions that were taken.

Further useful observations can be made on the effect of new hardware or software technology. This creates more alternatives and should lead to more effective systems.

All case studies from manufacturing companies will be difficult because of the wide range of products. The manufacturing process will vary from 'one off', through discrete batch to flow line production. Bonas assemble to order from a standard product range and manufacture components in batches to a forecast. The company was therefore a good fit for the typical application package designed for engineering companies.

2.0 Case Study

Bonas Machine Company Ltd is a privately owned engineering company employing 300 people in the north east of England. The company manufactures a range of textile machines for export throughout the world. The case study covers the selection and installation of a computerised Material Planning System during the period February 1978 to April 1983.

The company produces a range of narrow fabric looms. These are assembled only to customer order but the components are manufactured or purchased to a sales forecast. The control problem arises from the complexity of the product and from the fact that each finished loom will be unique to that customer. The market demands that a loom be supplied within a few weeks (or days!) of the customer placing the order.

During 1977 the range of models was increasing and stock levels rising alarmingly. Even extra staff in Stock Control would not have been able to solve the problem and a computer was needed to provide the fast reaction to changing customer requirements. The company had little experience of computers and decided to recruit a Systems Manager (the author). Most of 1978 was taken up by system selection and the choice of the right hardware and software.

The selection criteria were:-

Pre-written software
Terminal based systems (using database)
and Hardware expansion

A Hewlett Packard solution was chosen for both hardware and software. The computer (HP3000) was installed in January 1979.

2.2 Historical Summary

In formulating the installation plan, the main objectives were to maximise benefits at each stage and for the users to gain experience of the computer systems prior to the installation of Stock Recording. Hence the plan was for the installation of Bill of Materials and Purchase Order Recording as the first two modules. A temporary VDU operator was employed to help transfer the data and this allowed labour operation times to be entered and Standard Product Costing installed at the same time as Stock Recording.

At the same time that the computer was installed, a second person was moved into the computer section. A member of Stock Control was transferred to be a 'Liaison Officer'. The job involved user training, a limited amount of data input, and some computer operation. This approach was so successful that a second Liaison Officer was appointed in October of 1979 to specialise on Accounting, Costing and Spare Parts systems.

By the fourth quarter of 1979, Product Costing of assemblies had been computerised and test runs of MRP were possible. The latter showed that the stock recording data was poor and several months were spent on a review of the stock recording paperwork. In the absence of perpetual inventory, the installation of MRP had to await the next annual stocktake in August 1980.

The installation plan called for diversification into many areas of the company and, as planned, a computer programmer was recruited in August 1980. Whilst production staff continued to improve the quality of their data, the computer staff (four people) were able to extend the systems into other areas.

Systems were developed for spare parts, commercial (contracts), purchase variances and bonus calculation (job cards). The first part of 1982 saw the installation of a payroll package and, later the same year, the installation of accounting packages for the sales, purchase and nominal ledgers.

By 1983 the data volumes had grown to 25,000 parts, 110,000 structures and 32,000 labour operations. MRP was run monthly for long term planning with 'what if' special runs on request.

The initial hardware configuration included five terminals and one disc of 120Mb (this was backed up to magnetic tape). At April 1983 the system had grown to twenty terminals and three 120Mb discs.

3.0 Observations

3.1 Compromises

Several 'short cuts' were needed for the installation to succeed. The company had a conflict between Parts Lists as organised by Stock Control and those produced by the Drawing Office. The Stock Control version was used since this was common to stores and assembly. However, this did not solve the continuing problem of Engineering Changes and the need to cross reference the two Parts Lists.

In order to simplify the installation, the first version of Materials Requirements Planning extended only down to finished components. The machine shops were considered as suppliers and works orders recorded as single line purchase orders.

The most significant simplification was the method used to provide component issues to the Assembly Shop. This reflected the previous manual system and the fact there had never been sufficient staff in stores or stock control to record the stock issues as they occurred. Stock issues were actioned in the computer system after the loom had been built and batches consolidated to give one stock issue per part for all the looms that had been built in the previous two weeks! In this way, an average of 12,000 stock issues per week were reduced to 1,000 transactions per week. This had to be a two-stage batch process since stock 'shortages' had to be identified and resolved prior to the stock issue transaction.

3.2 Key Policy Decisions

3.2.1. Preparation prior to computer delivery

In 1978, during the computer selection process, a separate project was launched to provide better information from the sales invoices for spare parts. This involved the use, once per month, of a computer bureau to provide the Cost of Sales and parts usage analysis.

In addition to the management information, the system provided a computer master file that could be converted to the HP3000 and also started the necessary education process in the use of computer systems. When the HP3000 was installed, it soon became operational with the automatic transfer of data from the bureau system for 10,000 parts. A stock evaluation, at standard cost, was possible from stocktaking slips, within a month of the system being delivered.

3.2.2. Quality of Data.

It was known that the manual systems contained poor data and a formal decision was taken not to delay computerisation pending a review of existing procedures. Although this is contrary to normal theory, deferring the computer systems would have allowed only marginal improvement in the data.

Some of the weak areas did not become apparent until the systems were installed but the project required double the estimated (or hoped for) time of one year.

3.2.3. Hardware Expansion

The demand from the users for more terminals and more systems proved that hardware expansion had been a very valid item to consider during system selection. If the systems are successful then the hardware expansion will be at a greater rate than planned.

3.2.4. Staffing Policy

Given the industry shortage of experienced computer staff, a policy of in-house training was adopted. This applied to the two 'Liaison Officers' who were existing employees and to the computer programmer who was recruited immediately after completing his college training. This policy produced continuity, stability and a degree of team spirit that would have otherwise been difficult to obtain.

4.0 Conclusions

4.1 New Technology

Terminal based computer systems can work in practice provided that the input responsibilities are clearly defined and if the terminal user can save time on calculations or document preparation to compensate for the extra time required to input data. At Bonas no new staff were employed to be VDU operators, existing staff were trained to use the new systems. The computer can be treated almost as a telephone exchange with a minimum of operational requirements.

The implication for computer staff is that any 'central' DP people must be able to perform a wider variety of functions, ranging from computer operating through user training and systems development to management consultancy. Computer 'General Practitioners' are needed rather than specialists. This is important for productivity and cost effectiveness in a small installation.

The new software tools, and products such as RAPID, help to make this staffing concept possible, especially if a 'prototyping' approach is used for system development. In addition to the saving in development time, the resulting systems will be of a higher standard and far more effective.

4.2. Project Management

The systems development manager must always look for benefits for the whole company in planning a given project or selecting the next project from a list of alternatives.

The user departments have to be persuaded to accept the most useful 80% of a new system in exchange for fast development and installation. A second generation system can then be produced after six or twelve months of using the prototype.

The project manager must reconcile the conflicting departmental requirements whilst maximising the short term and long term benefits for the whole company. A project or system for just one department is acceptable only if there are short term gains and if the project is consistent with the long-term plan.

INTEGRATED SYSTEMS FOR MANUFACTURING COMPANIES

Presented by

P. Robinson

October 1983

Hoskyns Group Limited,
Springfield House,
Springfield Road,
Sale,
Cheshire.
M33 1XS

Tel: 061-969-3611
Telex; 665169

Offices in London, Birmingham, Manchester
Sydney, Dubai, Singapore, Johannesburg
U.S.A. (Martin Marietta Data Systems)

INTRODUCTION

MAS-II is a fully-integrated set of on-line application systems developed to meet the needs of manufacturing, industrial and commercial business.

It is built specifically to run on Hewlett-Packard 3000 Series machines, and is based on over ten years of experience installing other MAS systems on IBM, ICL, and Univac machines. Over 2400 systems have been implemented throughout the world.

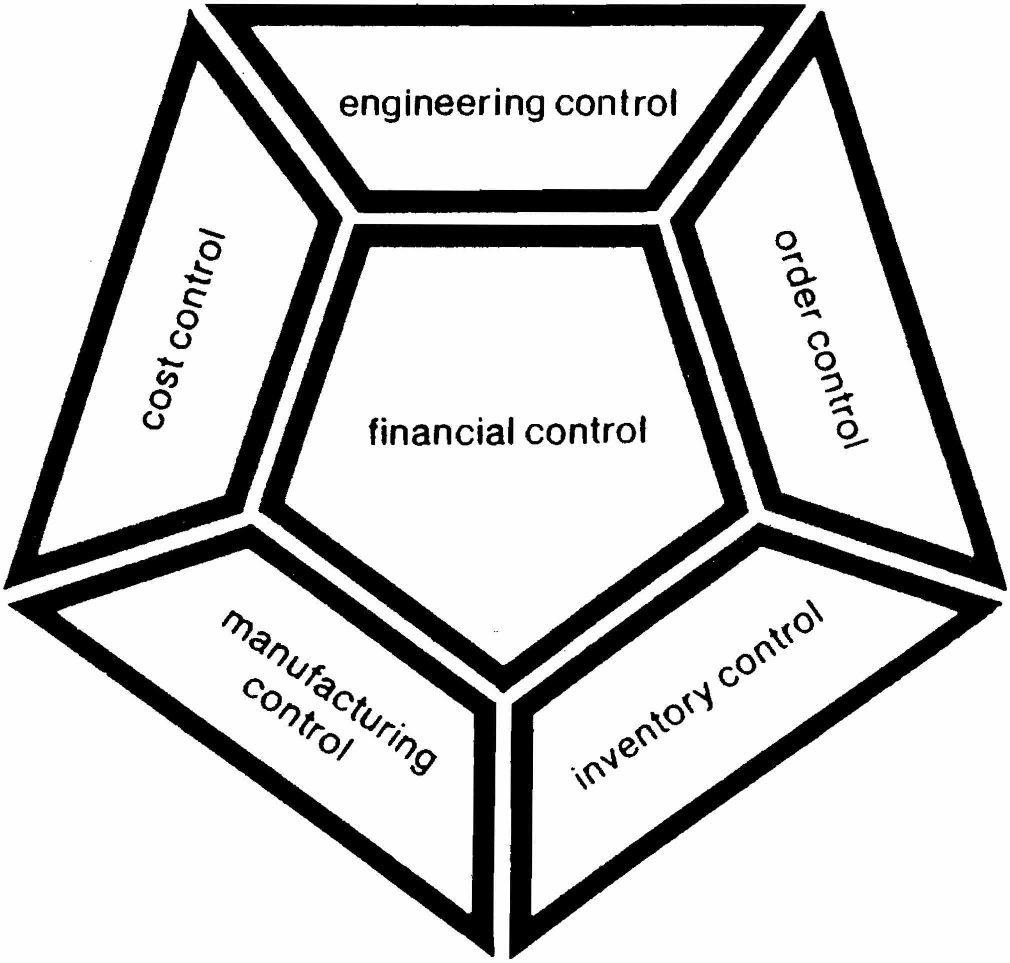
MAS-II consists of six basic modules. These are:-

- Engineering Control
- Inventory Control and Requirements Planning
- Manufacturing Control
- Cost Control
- Financial Control

These systems, and the benefits which they can bring to you and your organisation, are described briefly in this paper.

In addition to discussing the content of MAS-II, case study details will be given to illustrate the benefits during the presentation.

modular application systems (MAS)


engineering control
cost control
order control
financial control
manufacturing control
inventory control

FEATURES

The primary design criterion of MAS-H was to make it easy and secure for you to use. Some of its more significant advantages and benefits are outlined below.

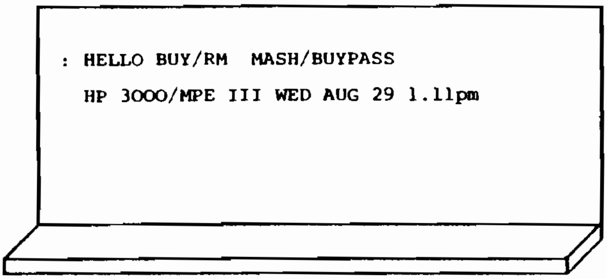
USER-ORIENTATION

Experience has shown that many computer systems fail because the user is forced to adapt his systems to the computer's needs. The flexibility of MAS-H means that the data entry procedures can usually be tailored, without the need for programming assistance, to reflect the user's current procedures.

Instead of grappling with the complexities of technical computer processes, the user is led step-by-step through the tasks he requires to perform on the VDU screen.

This simple user-orientated approach allows the input and retrieval of information to be made the responsibility of the user to whom that information belongs. Putting the power of the computer on the user's desk enables him to check and correct his own errors and to make his decisions on an accurate and timely basis, because information for those decisions is available when and where it is needed.

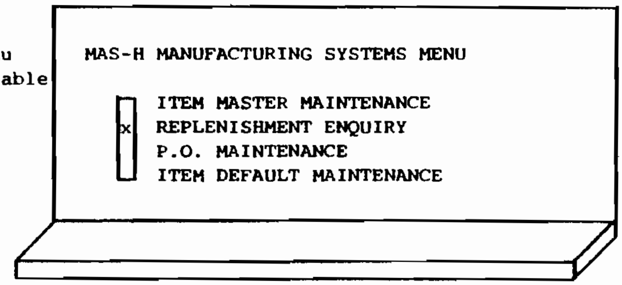
ease of use



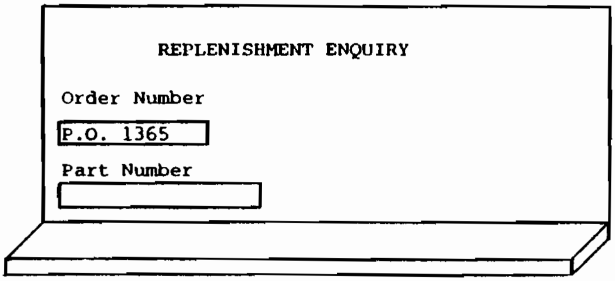
```
: HELLO BUY/RM  MASH/BUYPASS
  HP 3000/MPE III WED AUG 29 1.11pm
```

User - a raw material buyer - signs on to system, entering password.

System displays menu of procedures available to buyer. Buyer selects enquiry.



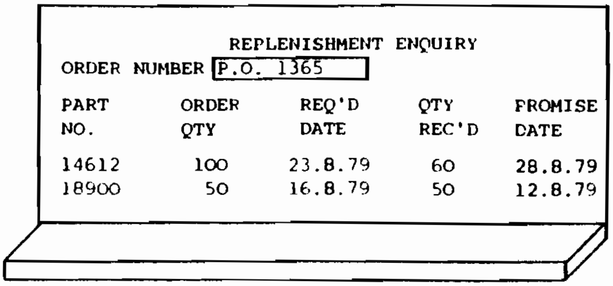
```
MAS-H MANUFACTURING SYSTEMS MENU

  ☐ ITEM MASTER MAINTENANCE
  ☒ REPLENISHMENT ENQUIRY
  ☐ P.O. MAINTENANCE
  ☐ ITEM DEFAULT MAINTENANCE
```



```
        REPLENISHMENT ENQUIRY

Order Number
[P.O. 1365    ]
Part Number
[                ]
```

Buyer enquires about status of items on P.O. No 1365

System displays details of order status.



```
              REPLENISHMENT ENQUIRY
ORDER NUMBER [P.O. 1365    ]

PART      ORDER     REQ'D      QTY      FROMISE
NO.       QTY       DATE       REC'D    DATE

14612      100      23.8.79     60      28.8.79
18900       50      16.8.79     50      12.8.79
```

FULL INTEGRATION

MAS-II comprises six modules covering all the major areas of management information and control:

- Engineering Control
- Inventory Control and Material Requirements Planning
- Manufacturing Control
- Cost Control
- Order Control
- Financial Control

All these modules are fully integrated. Data is entered at one point only. The fully integrated data base ensures that the information is immediately available to all other systems.

MODULARITY

MAS-II has been built with ease of installation and implementation in mind. The six modules are in turn composed of elements, enabling you to pick the systems you require for implementation. Apart from some logical dependencies all the systems may be installed on a stand-alone basis and in any sequence. For example you may start with Sales Ledger and work back into Sales Order Processing safe in the knowledge that invoices can be posted into the Sales Ledger system either manually or automatically via the Sales Order Processing System.

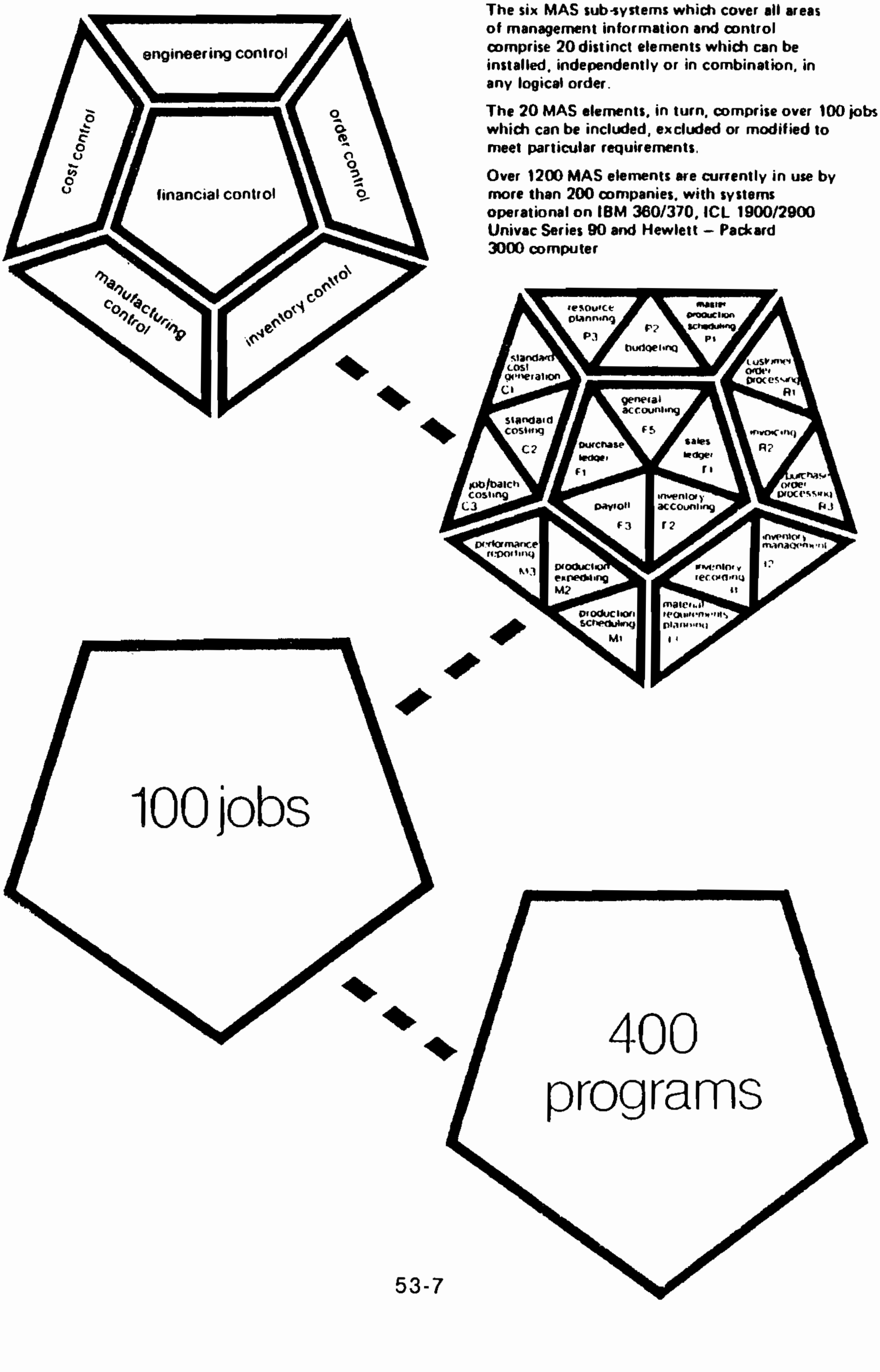
The ability to choose the starting point you want enables you to achieve tangible benefits more quickly than if a predetermined implementation sequence were imposed on you.

modular application systems (MAS)

The six MAS sub-systems which cover all areas of management information and control comprise 20 distinct elements which can be installed, independently or in combination, in any logical order.

The 20 MAS elements, in turn, comprise over 100 jobs which can be included, excluded or modified to meet particular requirements.

Over 1200 MAS elements are currently in use by more than 200 companies, with systems operational on IBM 360/370, ICL 1900/2900 Univac Series 90 and Hewlett – Packard 3000 computer


engineering control
cost control
order control
financial control
manufacturing control
inventory control
resource planning P3
P2 budgeting
master production scheduling P1
standard cost generation C1
customer order processing R1
general accounting F5
standard costing C2
purchase ledger F1
sales ledger F4
invoicing R2
job/batch costing C3
payroll F3
inventory accounting F2
purchase order processing R3
performance reporting M3
production expediting M2
production scheduling M1
inventory recording I1
inventory management I2
material requirements planning I3
100 jobs
400 programs

The system you need is configured by selecting the elements you require in order to meet your initial objectives. As your experience with MAS grows you can switch in new facilities, add additional controls and tune the system to meet new needs, confident in the knowledge that the interfaces between jobs are proven.

Moving step-by-step in a controlled fashion ensures the success of your project.

ON-LINE CAPABILITY

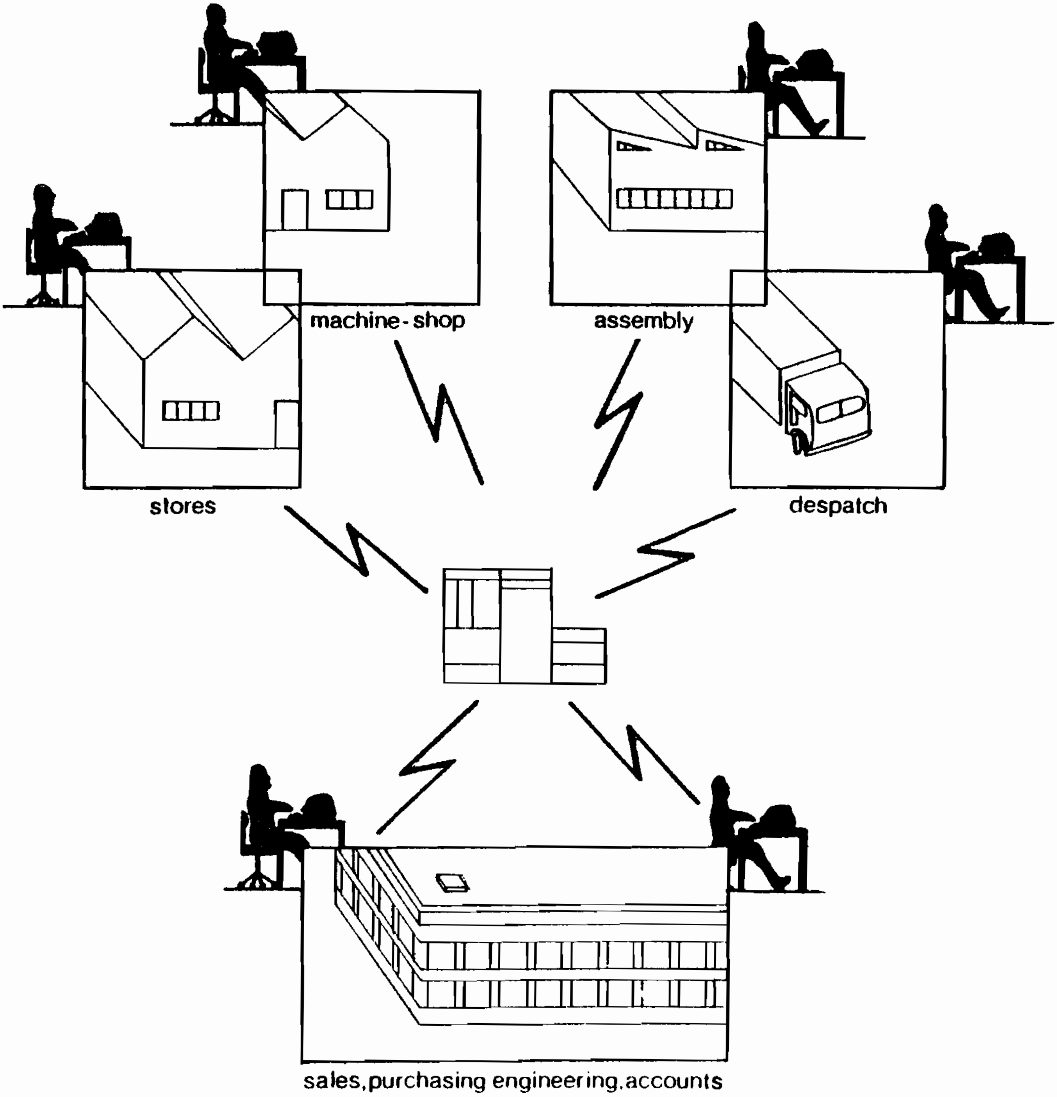
MAS-H brings you the benefits of the latest technology by enabling information to be entered into or retrieved from the system as soon as it becomes available.

The major advantages of on-line processing are:

Accuracy. It is almost impossible for a user to make errors unknowingly. This is because at the moment of attempting to enter or retrieve information the system checks the data base as well as carrying out the normal editing and consistency checks.

Timeliness. Because the user enters his own information to suit his own needs, information is always available on the most timely basis. The long series of tasks inherent in batch processing is avoided - document batching, input preparation, data control, error reporting, error correction.

distributed on-line processing


machine-shop
assembly
stores
despatch
sales, purchasing engineering, accounts

To these benefits MAS-H adds:

Flexibility. Most implementation problems stem from trying to change the company's systems to suit new computer procedures. The ability to change the format and sequence of data on input screens and to incorporate company-specific edit checks without the need for program changes obviates many of these problems.

Ease of Operation. Under the control of the systems manager, users gain access to the procedures available to them via 'menus'.

The need for training is kept to a minimum through this approach which leads the user through the procedures which are in every-day use for him.

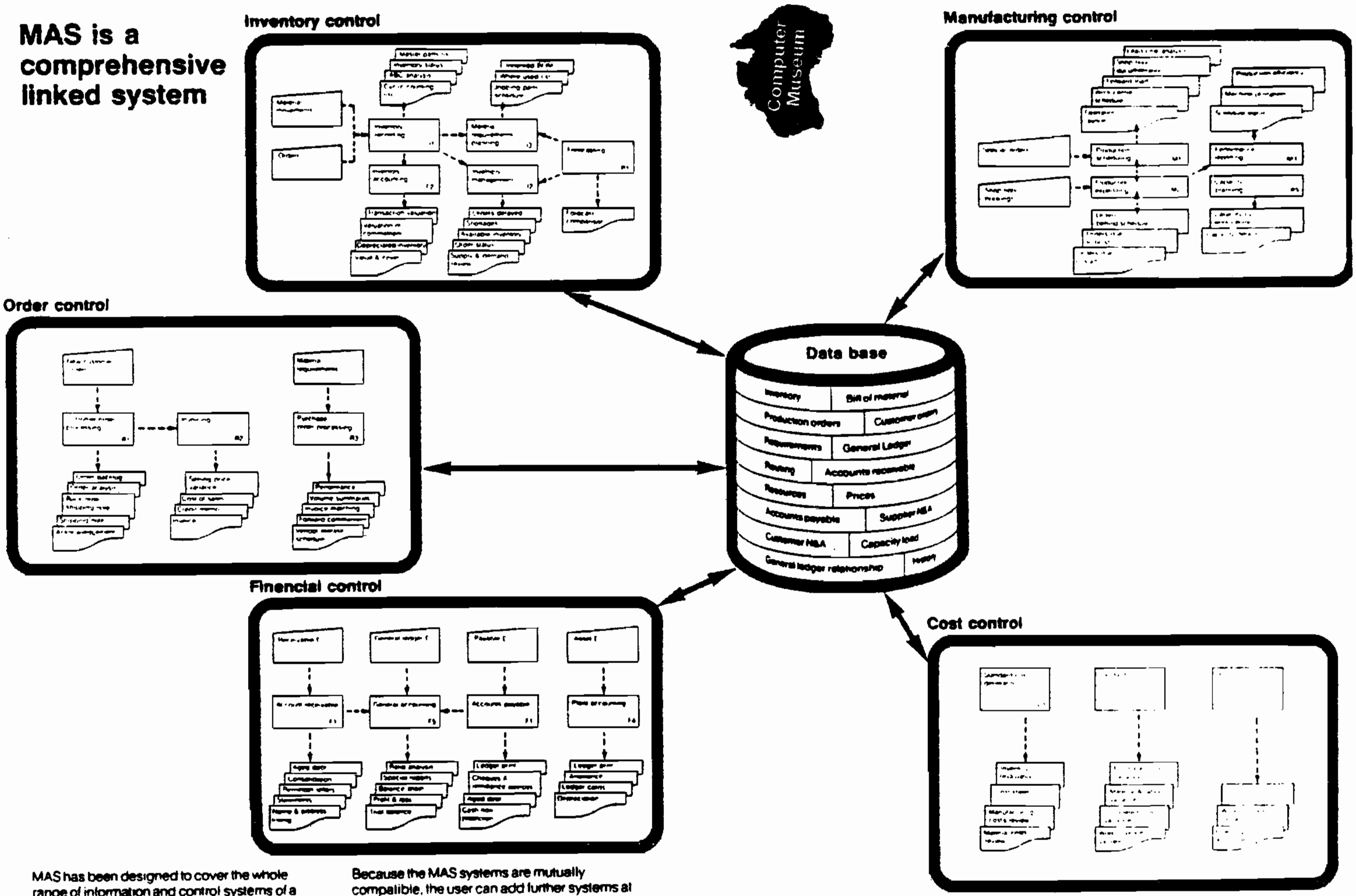
Privacy. The systems manager can arrange for certain procedures to be available to specific users only, and for data in the system to be available on a similarly selective basis.

Security. Comprehensive back-up and logging facilities allow the system to be restored without the necessity for re-entering data.

DATA-BASE FACILITY

The fully integrated MAS-H data base is implemented under Hewlett-Packards data base management system, IMAGE. It brings you these advantages:

MAS is a comprehensive linked system


Inventory control
Manufacturing control
Order control
Data base
Inventory
Bill of material
Production orders
Customer orders
Requirements
General Ledger
Routing
Accounts receivable
Resources
Prices
Accounts payable
Supplier N&A
Customer N&A
Capacity load
General ledger relationship
History
Financial control
Cost control
Computer Museum

MAS has been designed to cover the whole range of information and control systems of a manufacturing company

But the systems and the data base are modular, and applications can be implemented individually, in the sequence most appropriate to the user

Because the MAS systems are mutually compatible, the user can add further systems at any stage, confident that they will integrate well with existing systems

MAS can provide the basis for a total system development plan – with low initial cost and assured performance

- data is recorded once only
- information is available on a consistent basis to all systems and all users
- ease of systems modification
- information may be retrieved easily and quickly via the use of QUERY.

EASE OF MODIFICATION

Although every company will have a high level of common needs in any particular systems area, all companies have systems needs specific to the way in which they conduct their business. MAS-H recognises this mixture of standard and specific needs by providing a sound basic framework of facilities. Onto this framework the specific additions or modifications to facilities can be easily fitted.

We provide you with source code, full technical documentation, test data and a comprehensive test guide. This plus the modular nature of the programs and the high standard of programming, will enable your own staff to carry out your modifications as required.

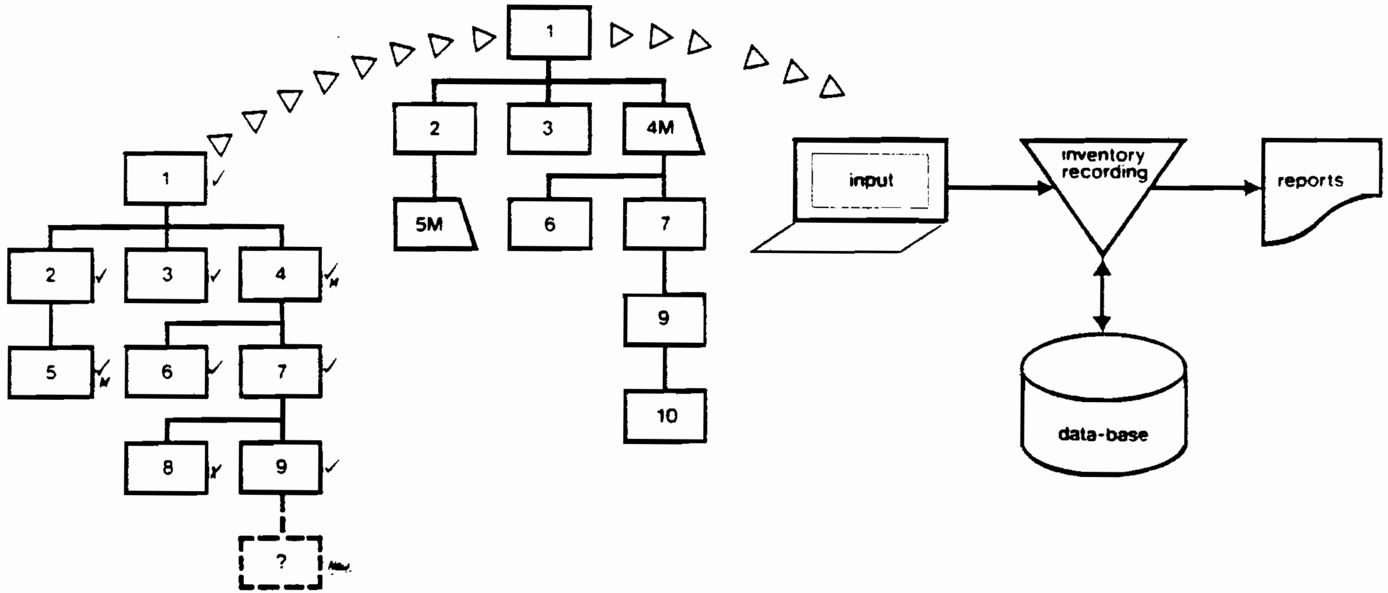
A wide range of support services is also available to meet your needs should you require them.

TAILORED SUPPORT

The MAS-H service consists of several major elements:

- advice and counsel from industry specialists in all areas to help you achieve the benefits you are seeking.

system building cycle


1
2
3
4
5
6
7
8
9
?
New
1
2
3
4M
5M
6
7
9
10
input
inventory recording
reports
data-base

take an MAS program,

and review the system specification. Decide which facilities (i.e. program modules) are wanted, which need to be modified, which are not required, and what new facilities are necessary.

carry out modifications,

remove unwanted modules, code and add additional modules and amend documentation. Because the programs have been designed for ease of modification, changes can be made quickly and accurately.

and you have your own tailor-made system

containing 80%-90% MAS code

- application software to provide a proven basis for systems implementation.
- modification support from experienced MAS-H staff to provide the tailoring needed to cope with any additional special system which you may require.
- training to promote understanding and acceptance of the system.
- technical support in the use of Hewlett-Packard hardware and software.
- software maintenance support from our MAS-H Product Centre to answer your queries, provide you with software updates etc. This is available on a phone-in or on-site basis on request.

The precise mix of these services will differ in almost every case with the needs of the company concerned - a further example of how MAS-H is tailored to meet your needs exactly.

TECHNICAL FEATURES

- COBOL. The use of the most widely accepted commercial programming computer language in the world gives you confidence in your ability to support the system in the future.
- IMAGE. Hewlett-Packard's database management system provides the database of information which is at the heart of the systems.
- VIEW. Provides you with a high level of flexibility in terms of being able to modify the system in one of the most critical areas - the data entry activity.

- Documentation. Full technical and user documentation is provided. This includes:

  - Installation Guide
  - Data-base Guide
  - On-line User Guide
  - Functional Systems Design
  - Computer System Design
  - System Reference Manual
  - Test Guide

The combination of systems which are:

  - user orientated
  - integrated
  - modular
  - on-line
  - supported
  - easily tailored
  - proven
  - economical

plus the commitment and experience of Hoskyns is the key to the success of your project.

ELEMENT DESCRIPTIONS

Brief descriptions of the major facilities offered by each system module, and an indication of the computer jobs available in each element of each module are contained on the following pages. Details of on-line and batch processes are described separately.

ENGINEERING CONTROL

Major Features

- On-line or batch maintenance of Inventory, Bill of Material, Routing, Work Centre, Tool, Process and Default Files, permits the building of a complete manufacturing data base.

- enquiries show details of:

    - all inventory items
    - product structure explosions
    - product structure implosions
    - routings
    - work-centre where-used
    - tool where-used
    - process where-used.

INVENTORY CONTROL

Major Features

- on line recording of all stock movements - issues, receipts, transfers, scrap etc.
- four order quantity policies, three safety stock policies, six forecasting policies.
- ABC analysis and reclassification.
- inventory value simulation
- full 'closed loop' logic.
- full visibility of supply/demand position available on-line.
- maintenance and enquiry on works orders, purchase orders, customer orders.
- net change MRP processing driven by Master Schedule or a combination of customer orders and forecast.
- exception reporting via trigger files for many conditions - potential shortages, orders due for release, overdue, order receipts, excess inventory.
- component availability checked before orders are released to manufacturing.
- full audit trail of all transactions.
- multiple unlimited stores locations.
- cyclic inventory reporting.
- valuation of inventory, forward commitment, customer order backlog.

MANUFACTURING CONTROL

Major Features

- schedules planned operations for orders ready for release to meet the due date.
- prints production documentation.
- handles labour bookings, scrap, work, for planned operations.
- on-line status reporting.
- provides performance reports on utilisation and efficiency.
- produces master production schedules and capacity plans.

COST CONTROL

Major Features

- handles frozen, current standard and new standard costs for material, labour direct overheads and indirect overheads.
- calculates new costs on the basis of the data contained in the Engineering Control data-base or amendments to that data base supplied specifically for this job.
- prints cost sheets.
- accumulates actual costs via Manufacturing Control, and reports variance from standard.
- values work-in-progress and scrap.

FINANCIAL CONTROL

Major Features

- Sales and Purchase Ledger systems maintain open-item or closed-item accounts for multiple companies.
- On-line entry and updating of ledger with batch control of input if required.
- On-line cash allocation and posting routines in Sales Ledger, payment control in Purchase Ledger.
- Statements, debt analysis, overdue letters, remittance advices, cheques, bank transfers.
- General Ledger with flexible chart-of-accounts.
- Report writer to allow the user to specify his own reports.
- Trial balance, profit and loss.
- Cost allocation, ratio analysis.
- Payroll provides up-to-gross, gross-to-net calculations.
- Department by department payroll preparation.
- Printing of all statutory documentation.
- Payroll analysis.

ORDER CONTROL

Major Features

- two separate systems maintain control of Purchase Orders and of Sales Orders.
- Purchasing system prints purchase documentation and maintains details of outstanding purchase orders.
- blanket orders and scheduled orders handled.
- supplier invoices validated.
- sales system checks stock availability at time of entering order, allocates stock to orders received.
- flexible pricing procedures allow special prices in customer/product combination, discounting at customer, item customer/item, product group levels. Manual price override permitted.
- items confirmed as despatched are passed through for invoicing.
- manifesting i.e., consolidation of orders for shipment.
- analysis reports and enquiries.

Using Plotters in Word Processing

by Rolf Frydenberg
Senior Consultant
A/S Fjerndata
Norway

Contents

Abstract
HP Plotters and HP-GL
Plotter Text Capabilities
Text Positioning
Font Selection and Generation
Editing the text
Conclusions

1. Abstract

Users of HP3000 computers have been asking for high-quality, flexible word processing output devices for many years. HP has answered by presenting the HP2680 Laser Printing System. But this device costs a lot more than most users are willing, or able, to pay for an output device.

But there is a solution available, and HP is the acknowledged world leader in that area: Plotters. All currant HP plotters are capable of handling text as well as lines, circles and other "ordinary" graphics.

This paper will present some of our experiences at Fjerndata, where we have been using an HP plotter for text output for about a year.

The paper concentrates on the practical side of using plotters for this purpose, particularily such areas as text size, fonts, etc.; as well as how to use this output device together with standard editors.

2. HP plotters and HP-GL

HP-GL is an acronym for Hewlett Packard Graphics Language. HP-GL is not a programming languaga as such, but a command language for manipulating semi-intelligent graphics devices, mostly plotters.

Most HP plotters can "understand" HP-GL, though there are some commands that are not implemented on certain plotters, and some commands that are not quite identical on all plotters. There are also some non-HP devices that can interpret HP-GL commands. These devices are not included in this presentation.

There are three major types of HP plotters: The HP7470, the HP7220 and the HP7580. These plotters vary in size, functions and HP-GL implementation. For the purposes of this paper we will ignore these differences between the three types.

Plotters may, form a user point of view, be distinguished through two basic measurements: The size of the plotting area (i.e. the size of the largest sheet of paper you can plot on at any one time), and the number of colours (i.e. pens) that you can handle in one pass.

Size of plotting area is normally measured according to the A-series of standard paper sizes. A-sizes go from A0 (A-zero) to as high as you want, since all sizes may be generated based on any other size. A1 is half of A0, A2 is half the size of A1, etc. A0 is 1 square meter, in the shape of 118 cm by 84 cm. You divide the long side by two to get A1, i.e. 59.5 by 84 cm; A2 is 59.5 by 42 cm; etc. Standard writing paper is A4, i.e. 29.7 by 21 cm.

The HP7470 is the smallest, and least expensive, of HP's plotters. It currently sells for about 1500 US dollars. It is also the smallest one, and has the smalles plotting area (A4). The HP7470 only has 2 pens available, so in order to produce multicolour output, you must at some point switch pens.

Current models of the HP7220 plotter has 8 pens, giving you up to 8 colours, or four colours in two pen widths. The HP7220 has a plotting area of A3, and can take paper rolls. It is therefore possible to geenrate very long 30 cm wide plot output on thisd printer. But it does not have any standard "scroll" function, but feeds one or half a page at a time. So to use the paper roll to exceed the A2 plot area, you must do your own special plotter control, and position calculations.

The HP7580 (and HP7585) are HP's large size plotters. The 7580 has an A1-size plotting area, while the 7585 can plot on sheets up to A0.

The HP-GL command language consists of approximately 80 instructins. There are three groups of commands that are of special important to us in text plotting. These groups are:

Vector Group. This group contains most of the commands used to move the pen around on the plotting area. These commands are also used to draw straight lines.

Arc and Circle Group. This group contains commands for drawing circles and arcs. The HP7470 plotter does not support these commands.

Character Group. These commands are used for generating text and characters. The most important commands in this group are supported by all HP-GL plotters.

There are also a large number of commands for utility funotions such as plotter configuration, status reporting, set up etc., and a group of commands for using the plotter as a digitizer. We will not cover these commands in this paper.

3. Plotter Text Capabilities

There are two primary methods for outputting text on a plotter. Both of these are quite common, but they are normally used in different situations. These two methods are:

- Plotting each component line of each letter separately (this would mean plotting three lines to generate an N).
- Use the "hard-wired" character set(s) in the plotter, and just transfer one character for each character you want to print.

For labelling drawings and other complicated graphical presentations, where the text is only a small part of the total plot, the first approach makes perfect sense. This approach can also allow you to use many different "fancy" type-fonts, such as the GOTHIC font supported by HP's Decision Support Graphics package, and other software.

When text is your primary output, it is more doubtful whether you can use the same approach in a sensible manner. In this type of situation, I prefer to use the second approach. This still allows you to use various type-fonts, but not as fancy ones as GOTHIC and so on. And it is a lot faster, too.

HP plotters draw text for you, if you use the Label (LB) command. The command consists of just the two characters LB, followed by the text you want plotted. You must use a special, pre-defined terminator to indicate the end of the LB command. Default terminator (and the one we use most of the time) is End-of-text (control-C). Whenever we refer to this character, we will write it as <ETX>.

Text is always output starting from the position at which the pen happens to be located at the moment when the LB-command is interpreted, using the "current" size and font, and with the pen currently in the pen holder. So prior to LB-commands, you must select the pen you want, position the pen on the paper, and set parameters that define the characters you want to output. We'll return to this in detail later.

One important point about the LB-command is that it can handle a large number of characters within a single command. The LB-command also executes any Carriage Return and Line Feed characters that it receives, so it can be used easily for whole paragraphs (or even a whole page) at a time. This simplifies output quite a lot.

4. Text Positioning

A plotter, even when used for text output, does not "emulate" a printer. Instead, the user must control the plotter through commands. This is valid for just about every aspect of text output on plotters.

Text output on a printer starts at the Top-Of-Form position, which is the upper left corner of the paper. The default position on a plotter, on the other hand, is the lower left hand corner, which is not the best of positions to start outputting text. So, even before you can start printing anything on the plotter, you have to learn how to move the plotter pen with commands.

Luckily, plotter pen movement is very straightforward on HP's plotters. Generally, there are two types of commands that move the pen: The Plot Absolute (PA), and the Plot Relative (PR) commands. The PA command moves the pen to a specified position in the plotting area, while the PR command moves to pen relative to where it is currently.

The PR command is more difficult to use than the PA command, since the plotter pen may move without your knowing exactly where to. Any text output moves the pen, so you must be extramely careful when moving the pen relative to its location after you have used it to write text. In this paper we will only present text positioning through the PA command.

HP plotters are addressed in so-called plotter-units, which are approximately 1/1000th of an inch. The exact measure is 0.0250 mm whereas 1/1000th of an inch is 0.0254 mm. The approximation of 1000 plotter-units to an inoh gives an error of 0.4 mm pr inch, or 1.8 %. This means that we have 1018 plotter units to an inch.

A normal A4 sheet of paper is 210 mm by 295 mm. In plotter-units, this is 8400 by 11800. This is the maximum erea available to plot in. For an american standard 8 by 11 inch paper, the available area is 8128 by 11178 plotter-units. But we cannot really fill all of this area with text. We need margins on all four sides of the sheet - say at least 500 plotter units (half an inch) . This reduces our text-plot area to the coordinates:

	A4	8 by 11
X-axis:	500 to 7900	500 to 8628
Y-axis:	500 to 11300	500 to 10878

For text output, we have developed our own set of standards, that describe where the text is supposed to be placed. We place the first line in position (750, 11000), and we leave a space of one blank line between separate paragraphs.

This paper has been plotted according to our standards, so you can see for yourself what they are.

Sometimes, we use some type of "frame" around the text pages, the way I have done it with this paper. We do this by using the line-drawing possibilities of the plotter, using HP-GL instead of standard graphic utilities. These pages were plotted using an HP7220T plotter, and the sequence of HP-GL commands to draw the frame was:

```
SP5; PA0, 1000; PD;
AR+500, 0, 90; PR+8200, 0; AR0, +500, 90; PR0, +9800;
AR-500, 0, 90; PR-8200, 0; AR0, -500, 90; PR0, -9800;
PU; SP0;
```

In addition, I have put the Fjerndata logo (an F) in the lower right-hand corner of each sheet. This F has also been generated by using HP-GL commands.

5. Font Selection and Generation

Character fonts on plotters are selected by specifying the size of the characters, and by slanting the characters. Both size and slant can be varied by using real values, which means that there is an all-but-infinite set of fonts that may be used. You can also generate bold fonts, by causing charcters to be overprinted, with a slight offset.

The character size is defined through the HP-GL character size (SI) command. This command has two parameters - width and height. The size parameters are addressed in centimeters (1 cm is 400 plotter units).

In addition to the size of the characters generated, the SI command also defines the amount of space the plotter will generate between characters (one half the width parameter) and the spacing between lines (equal to the height parameter).

In most cases, you don't really need a lot of different sizes. So we have to some extent standardized, and most commonly use two sizes, one for headings, and one for "ordinary" text. You can see which sizes they appear to be in this paper. The selected character sizes are:

Headings:	0.3 cm wide, 0.45 cm high.
Text:	0.16 cm wide, 0.22 cm high.

For italics, you may slant the characters, by using the Slant (SL) command. This command will cause all subsequent text to be slanted, until a new SL command resets the previous one. In the SL command you specify the tangent to the angle between the vertical axis and the uprights of the characters. For slanting this paragraph, we have used a tangent value of 0.3.

The second manner in which you can generate special character fonts is by double-printing the text, with a slight offset between where the text is positioned each time. This generates BOLD characters. This paragraph was printed by using an offset of 5 plotter-units on both axes.

6. Editing the text

The text that you want to plot can be entered into the computer by mweans of any editor. I usually use HPSLATE for this type of text input, especially because of its easy text-justification capabilities. Because HPSLATE uses a very special file layout, I convert the finished file to a standard EDITOR-type file before plotting it.

But you can use any editor to edit the text that you want plotted - TDP, QEDIT, SPEEDEDIT or any of the other ones available. If you use an editor that uses a special file format (as QEDIT does), you will have to convert to a normal EDITOR-type file before you can start preparing for outputting the text on the plotter.

When you have entered your text, and have converted it to EDITOR-type file format, the next step is to prepare it for plotter output. I prefer to split the text into page-sized files before I output it on the plotter. This is also more compatible with the way in which the plotters work - they generally only output a page at a time. These files are work-files, into which I copy a part of the text file. I purge them after I have used them.

The page-sized files can now be manipulated to produce files that can be output directly on the plotter. This means inserting HP-GL commands into the files. I normally do this in HP's standard editor - EDIT/3000. And, of course, these files are UNNUMBERED files. They should contain 80-byte records, for compatibility with the definition of the plotter in the MPE I/O configuration.

The simplest type of plot-fila contains only the following commands:

1. Initialize: Select pen, position to top left-hand corner (e.g. SP1; PA1000, 10000; - Select pen #1, Start output at position 1000, 10000).

2. Plot the whole page (e.g. SI0.5, 0.375; LB ...text... <ETX>; where ...text... may span multiple lines).

The next step in sophistication, is to print the heading in a different size. You do this by using additional SI (character SIze) and PA (Plot Absolute) commands. E.g. The headings in this document where generated through the following series of HP-GL commands:

```
SP5; SI0.3, 0.45; PA750, 11000;
LB ...heading...
<ETX>;
```

This causes the heading to be positioned from the point 750, 11000, and using an extra line for the terminator (<ETX>) causes the plotter to execute a carriage return and line feed. The plotter pen will now be at a position directly below the heading.

Starting at this position, the text is plotted using the command sequence:

```
SP1; SI0.18, 0.22; LB
...text...
<ETX>; SP0;
```

By placing the text on separate lines it is a lot easier to edit the text, then if text and commands were on the same lines. I have also included a pen selection command in both the heading output sequence and the text output sequence. The reason for doing this, is to ensure that text is always plotted in the right colour and line width, even when you have changed the pen selection in one command.

The actual text plotting is accomplished by using the standard MPE file copy utility FCOPY. You do this by using the following MPE commands:

```
FILE PLOTTER; DEV=xx

FCOPY FROM=plottext;TO=*PLOTTER
```

We normally plot out a file that contains initialization commands for the plotter first, and a plotter termination file afterwards. This ensures that the plotter is available for access from other applications, after we have finished using it for text plotting. It is important to do this if the plotter is connected to a terminal through an eavesdrop cable - the plotter will not let data through to or from the terminal if it hasn't been "turned off" by a command.

7. Conclusions

In this paper we have seen how an HP plotter can be used for printing text. It is now time to look back, and answer the question I think many of you have been asking yourselves: WHY output text on a plotter?

I think there are severel good reasons for using a plotter as a text output medium. The main ones are:

> The plotter gives a much nicer-looking document than an ordinary letter-quality printer gives you. Its ability to use varied text sizes is a major advantage.
>
> Outputting text in the manner we have described here, is normally much faster than the text output functions of most graphics packages, but you have to sacrifice typefonts such as Gotohic and Roman.
>
> It is a lot cheaper to output text in this manner than on a laser printer, which is able to do the same type of character scaling that we do on the plotter.
>
> Compared to graphics systems, using the HP-GL approach, puts a much smaller load on the system. The CPU-consumption of graphics programs is very high in general.

THE PAINS & PLEASURES OF OFFICE AUTOMATION THE HPWAY -

A CASE HISTORY OF OFFICE AUTOMATION AT GLAXO PHARMACEUTICALS

by

M.H.Wadsworth,
Glaxo Pharmaceuticals Ltd.,
Greenford, Middlesex.

Summary

Glaxo Pharmaceuticals have a network of seventeen HP3000 minicomputers with over five hundred terminals attached. The network is spread throughout Great Britain and serves a dozen different sites.

The way in which Office Automation products have been introduced to the network together with the troubles and benefits arising from the adoption of the Hewlett Packard approach is dicussed. The products used include HPMAIL, HPWORD, WORD125, HPSLATE, HPTELEX, DSG and EASYCHART.

The use of non HP products for financial modelling and interaction with Prestel are outlined together with hopes for future development.

Introduction.

Glaxo, the largest British Pharmaceutical Company, has factories and subsidiaries in over fifty countries around the world and is one of the world's top twenty pharmaceutical concerns.

It deals largely in "ethical pharhmaceuticals" (prescription medicines) and is currently best known for it's anti-ulcer drug ZANTAC, it's anti-asthmatic drugs VENTOLIN and BECOTIDE and it's antibiotics CEPOREX and ZINACEF.

In the United Kingdom Glaxo has over a dozen sites spread throughout the country from Montrose in the North East of Scotland to the London area. The company therefore requires rapid and efficient methods of communication.

In 1978 the pattern for future Office Automation was set by the decision to establish two communications networks. It was decided to develop a distributed communications network based on HP3000 minicomputers to provide the Data Processing requirements of the whole company. The second network was a telecommunications network eventually to be composed of private digital exchanges. The development of the computer network together with it's DP systems and the consequent terminals and other peripherals has progressed rapidly and now consists of 17 HP3000s serving 12 sites. Over 500 terminals are installed on the network and provide an excellent infrastructure for our Integrated Office Automation Strategy.

Although it had been foreseen in 1978 that the policy of a distributed Computer Network would provide the springboard to Office Automation it was not until the early 1980s that the necessary sofware packages began to appear.

In the first half of 1981 we entered the field with an examination of Decision Support Graphics and a Visicalc Package for Financial Modelling. The latter package proved inadequate and was replaced by the much more comprehensive FCS-EPS system.

Later the same year we experimented with HP's version of Spellbinder, WORD125, running on their 125 "Business Assistant" microcomputer.

Electronic Mail appeared on the scene the following year when we undertook a Beta test of HPMAIL for Hewlett Packard.

Our introduction to Hewlett Packard's own word processing software came about the same time, but difficulty in obtaining hardware - not an isolated problem - delayed it's introduction until a few months later.

Finally computerised Telex, once again delayed by problems of hardware delivery, was installed in May of this year.

I will now discuss the experiences we have had with these products and how we hope both the products and Office Automation at Glaxo will develop in the future.

HPMAIL

As a Beta test site in February 1982 we had a chance to evaluate HP's electronic mail at a very early stage. The system was set up on seven computers over six sites with about twenty five users all from within the Management Services Department.

The product, although it had several minor teething problems, proved to be very reliable and able to recover from machine failures without either major effort or loss of mail. Consequently we agreed to purchase the system at the end of the trial and began to extend the number of users on the system.

One of the justifications for the purchase of the system was it's ability to transfer software developed at Greenford to computers all round the network. After trial transmissions this was initiated on a live basis in July 1982 and is estimated to save the work of two people transmitting the software manually.

The system continued to expand and in March 1983 users were set up on a further three HP3000s.

In May of this year a British Telecom DCE1 modem was installed on one of the computers at Greenford releasing HPMAIL from the confines of our network and allowing it to reach any HP3000 with a synchronous link to the Public Telephone Network.

Finally in June we installed HPDESKMANAGER, the enhanced version of HPMAIL, on a non-networked linked machine for evaluation. The immediate problem we encountered was the hole it made in the security of the system. Within the Work Area it is possible to issue MPE commands, including the RUN command, overriding the restrictions imposed by log-on UDCs. This problem has yet to be overcome and to date the system has only been installed on our three development machines.

The way we expanded the system following the Beta test was to select groups of users who both had the need to communicate, preferably on a multi-site basis, and were well equipped with terminals. We have made it a general rule that we do not provide new terminals to people purely for the use of electronic mail. We have found that HPMAIL is very popular and effective when a user finds that all the people with whom he communicates are also HPMAIL users. In a communicating group where half or more are not HPMAIL users the system is not popular and it's use declines.

Using these criteria the number of users on the system has gradually increased from the initial twenty five of the Beta test to a current level of virtually four hundred users now spread over fifteen computers and twelve sites. The last few months have also brought us to the situation where people are asking to be put onto HPMAIL in order to communicate with existing user groups.

Let us now examine the strengths and weaknesses of HPMAIL. Firstly it's strengths:

It is a "user friendly" system being relatively simple to use and containing very good inbuilt "help" facilities. The self paced training package which comes with the system, whilst having proved useful for a few people, is rather longwinded and most users tend to find it is easier to learn by using the system.

The system is both reliable and robust and we have very few problems even with about 400 users. System failures do not cause any serious errors and the worst problems arise from databases becoming full, as I will mention later. Occasional problems are the stopping of the transport mechanism either due to a MAILTRUCK becoming stuck on a DS Line or the aborting of the transport programs.

Both these occurrences are easily rectified by closing down and restarting the transport system which is done without affecting system users. The final intermittent problem is that of users being inexplicably ejected from the system with a message to the effect the "HPMAIL has encountered a serious error". This is nearly always not repeated when the user signs on anew and repeats the process he was engaged in.

The ability to put acknowledgement levels on the mail you send is a very valuable feature of HPMAIL. It not only enables you to know that the system has safely delivered your mail but also allows you to tell if a particularly important message has been read - this also provides a method of checking those users who don't read their mail regularly.

Finally it can be used on any terminal connected to an HP3000. This is an important feature for any Office Automation system and has enabled us to use an Exxon 520 Word Processor as a terminal for acessing HPMAIL.

Next we have the shortcomings of HPMAIL

Firstly the problem that has caused us most trouble in the transport of software and large forms files around the system is "inefficient networking". By this I mean the HPMAIL decides how to package messages to multiple destinations throughout the network. For example sending a message from the development computer Greenford C to the five live machine Montrose, Ulverston, Barnard Castle 1, Speke 1 and Ware 1 would involve sending five copies to Greenford A, to Greenford 1 and to Greenford 2 before they split up to their respective destinations. As you can well imagine if this message were a forms file of some 3000 sectors the databases of the intermediate machines would rapidly fill up resulting in crashes of the system and in some cases loss of mail. To minimise this duplication of messages we have set up dummy users at the hubs of the network and employ HPMAIL's AUTOFORWARD facility to send the messages on to their final destination.

Another problem, which has been resolved with HPDESKMANAGER, was the short message length that was permitted for the entry and editing of messages. The length is determined by the memory of the terminal being used which and further frustrations for users who regularly use terminals with different memory sizes.

The final problem I have listed is database locking. The method of locking employed by HPMAIL is not efficient and although this is not a problem for normal messages considerable delays do occur when software is being copied into or out of the system.

WORD PROCESSING

I will now move on to discuss the use of Word Processors within Glaxo Pharmaceuticals.

Our introduction to Word Processors was, like a lot of people's, with the Vydec - a machine which has proved to be extremely popular with it's users and, indeed, we still not only have several still operating but also have users asking for more!

Early in 1981 we experimented with WORD125 which is Spellbinder adapted by HP to run on the HP125 and the package makes very good use of the micro's programmable soft keys. The system requires the operator to remember more control codes than most modern WP systems and the editing features are not as flexible, particularly in the adjustment of tabulation settings. However it is a reasonable word processing system and we recommend it for users where the benefits of a stand alone microcomputer can be made use of in other areas. The notable example is in our Accounts Departments where the HP125 can offer the facility of the modelling package micro EPS which I will mention later.

About the same time as the introduction of the HP125 we also installed some Exxon 520 Word Processors. Here the demand for some form of Word Processing could not be deferred until the advent of HPWORD, which we were assured was just around the corner. The Exxon 520s were chosen because they ostensibly came from the same stable as the Vydec and were claimed to be linkable to the HP3000 computer. After one or two initial problems with the training of users the systems have become popular with their users and have not caused a lot of problems.

Finally we have HPWORD which was installed within the Management Services Department about the middle of last year, and has gradually spread from there. Initially the hardware was difficult to obtain - a position which for one reason or another does not seem to have changed very much ever since.

HPWORD I was limited by the need for two computer ports per system, one for the terminal and the second for the printer, a fatal flaw on an already crowded network, and so only a couple of installations of this version were made. With the advent of HPWORD II and the capability of the printer to be supported from the terminal we were able to expand the number of users without filling up all the computer ports, or at least only filling them at half the previous rate. We now have about twenty five systems operational and all are working satisfactorily.

Let us now look at the good points of HPWORD:

Compatibility; HPWORD operates from an HP2626W terminal which is a modified version of HP's top of the range non-graphics terminal and, as such, is capable of running any program which does not require a graphics terminal. Also the software is compatible with other HP systems; for example QUERY can be used to extract data from an IMAGE database and store it in a form that can be included in an HPWORD document. We have used this technique for sending a personalised letter to each of our suppliers using the names and addresses stored in an image database. This compatibility is undoubtedly the deciding factor in the choice of HPWORD as our number one word processing system.

The risk of loss of work in the event of machine failure is minimised by the storage of text by paragraphs. The maximum loss can only be a paragraph as opposed to a stand-alone system where the loss can be several hours work if storage is not carried out more regularly.

The system is easy to use and has the good editing facilities that one would expect from a modern word processing system.

One feature of HPWORD II that has proved useful is the ability to cater for non-standard daisy wheels by the use of specially created environment files; it is surprising how manyusers have a favorite typeface which can only be obtained on non-standard daisy wheels or who require a character which can be obtained by a double strike of two different petals of the daisy wheel.

Now we have the shortcomings of HPWORD:

Undoubtedly the major shortcoming is the fact that HPWORD requires the HP3000 to be running and available to users for word processing to take place. In our case this immediately restricts use to the hours between 7.30 a.m. and 6.00 p.m., a restriction which is not acceptable to all potential users.

The next shortcoming is, perhaps, an unwarrented criticism as it is difficult to see how the full features of HPWORD could be offered on a standard terminal. However the fact that a special and expensive terminal is required is undoubtedly a disadvantage to the installation of new systems.

HP did not initially provide training courses for users but only for those supervising the systems. Instead they provide a self-paced training package for use by the new secretary on her own equipment. The disadvantage of this method is that it is difficult either for the secretary to undergo the training at her own desk due to the intrusion of the other facets of her job or, alternatively, to set up the equipment in a special area for training. I understand that HP have just modified this policy and are now carrying out a three day training course, although the 390 pounds they charge does put off several new users!

HPWORD is not good for filling in forms. Certain forms can be catered for if templates are designed, but this is time consuming and will not cope with all forms.

One criticism I receive from users of HPWORD is troublesome pagination. HPWORD will automatically paginate avoiding widows and orphans, but will quite happily end a page immediately below a heading or in the middle of a table. Page endings can be inserted manually or lines can be "bound" together but either of these procedures is time consuming and fiddly. The main problem arises because there is no indication when you are actually typing as to the number of lines you have typed on the given page.

We would hope to see HPWORD develop the following features over the next few years:

Firstly it should become a stand alone system to enable document entry to be independent of the availability of the HP3000 but to enable full compatibility with the HP3000 to allow document transfer etc to take place.

The system should have the ability to include graphics, and preferably to be able to run on a graphics terminal.

HP should provide training courses at reasonable prices.

The system should include a much improved forms filling capability.

Output facilities should become faster, quieter and cheaper than the daisy wheel printers now available.

HPTELEX

HPTELEX was installed in May of this year after several months of waiting for both British Telecom to install the necessary DCE 3 modem and for HP to deliver their Telex unit. On installation our DCE 3 proved defective and it was a further week or so before the system began to work properly.

Our use of the system to date have revealed the following good points for HPTELEX:

The system is simple to use for sending single page Telexes; after only a few minutes instructions most users can happily send off their own Telexes.

The large Directory of Telex numbers that can be stored by the system save the user from having to remember the codes and numbers for regularly used destinations.

Telexes can be sent from any terminal which can support block mode making the system available to virtually all of the terminals attached to our network.

The Telex queue makes it possible for all users to check that their Telexes have been transmitted correctly.

The weak points of HPTELEX which have been revealed to date are:

The system has a high demand on computer ports, requiring one for the Telex printer and two for each Telex line used on the system.

The sending of Telexes of greater length than one page is complicated by the fact that an ASCII file of the message must be created on the system first. This requires the user to have access with a second facility such as HPWORD or HPSLATE and involves some form of file conversion.

The Directory, mentioned earlier, is displayed in a totally random fashion making it difficult to dicover whether an addressee of interest is on it.

The use of the programmable softkeys has not been as well designed as in other HP office systems. An inadvertant press of the SEND key after typing in one's Telex can commit it irrevocably to the ether rather than to the Telex line.

The system cannot be operated by character mode terminals and so is not available to devices such as an Exxon 520.

We would like to see HPTELEX develop the following features in the not too distant future:

An improved method of dealing with incoming Telexes, perhaps an interface with HPMAIL, is desirable to help overcome the delays and problems involved in distributing Telexes from a centralised printer.

Programmatic analysis of the Telexlog to enable reconciliation of the Telex bill would be an extremely useful enhancement. Considerable time and effort is currently spent doing this manually.

The use of the programmable soft keys should be adjusted so that it is not possible to inadvertantly delete one's Telex before transmission

The Directory should be displayed in alphabetical order and it should be possible to access the directory after entering the Telex message.

GRAPHICS PACKAGES

We have examined several Graphics packages from Hewlett Packard and are currently using three:

The packages designed for the 2647, Auto- or Multiplot and the Graphics Presentation Pack are used within the Management Services Department for the presentation of monthly statistics and in the preparation of reports. The systems are particularly good for producing textual slides.

DSG3000 the first of HP's 3000 based Graphics Packages is used within our Accounts Department for producing financial graphs for inclusion in their monthly reports. The system is versatile but is slower to use than the 2647 based systems particularly for textual slides.

EASYCHART, a more recent HP3000 package, is very simple a quick to use but has limited choice of format and does not cater for textual slides. It is currently being tested as a means of producing production statistics at one of our Secondary Production Sites.

HPDRAW, the top of the range product for use on the HP3000, can be used to combine and manipulate charts from DSG and EASYCHART and has good facilities for adding text. However we have not found an application for this system which justifies it's large usage of computer resources.

GRAPHICS125 has been examined and appears to be similar to the multiplot package on the 2647. Currently we have no users of HP125s who have requested a graphics package.

DSG and EASYCHART can both be used from any block mode terminal and so together with the HP7470A two pen plotter, at around 1000 pounds, can provide a relatively cheap graphics capability.

OTHER SYSTEMS

Finally I will discuss the various additional features that we either have on our system or are in the process of investigating:

Financial Modelling systems have been used at Glaxo for several years since early 1981 when Visicalc on an HP85 microcomputer was investigated. This was quickly found to be very limited in it's capacity and capabilities and we moved on to FCS-EPS a much more comprehensive package from EPS Consultants, the HP3000 version of which had been developed in conjunction with our colleagues at Glaxo Holdings. This has proved very successful and we have recently purchased it's subset Micro FCS for use on the HP125. It is envisaged that our Production Sites will use Micro FCS to produce their site reports which can then be transmitted to Greenford and combined and refined using the HP3000 based system.

The possibilities of accessing Prestel from any terminal connected to the computer network is currently under investigation. The system which is designed by Zycor Ltd and based on their Teledec TI interface unit, is being successfully used elsewhere and our initial investigations look hopeful. The system filters out the graphic and colour features and displays the remaining data on the terminal screen.

Access to other external databases would be another enhancement to our system which we hope to obtain in the near future, either as a spin-off from the Zycor system or from use of an autodial modem.

Glaxo is currently examining the use of a fibre optics LAN to connect terminals and computers on our Greenford site. It is hoped that this system will enable us to locate our office systems largely on one computer and thereby minimise on the costs of replicating software. This system would also provide a more efficient means of offering such services as HPTELEX to the maximum number of users.

I hope that I have managed to provide some insight into the way in which we introduced Office Automation at Glaxo and have indicated some of the pleasures of doing it the HP way without dwelling too long on the pains!

1978 – SETTING THE PATTERN

A. A DISTRIBUTED COMPUTER NETWORK

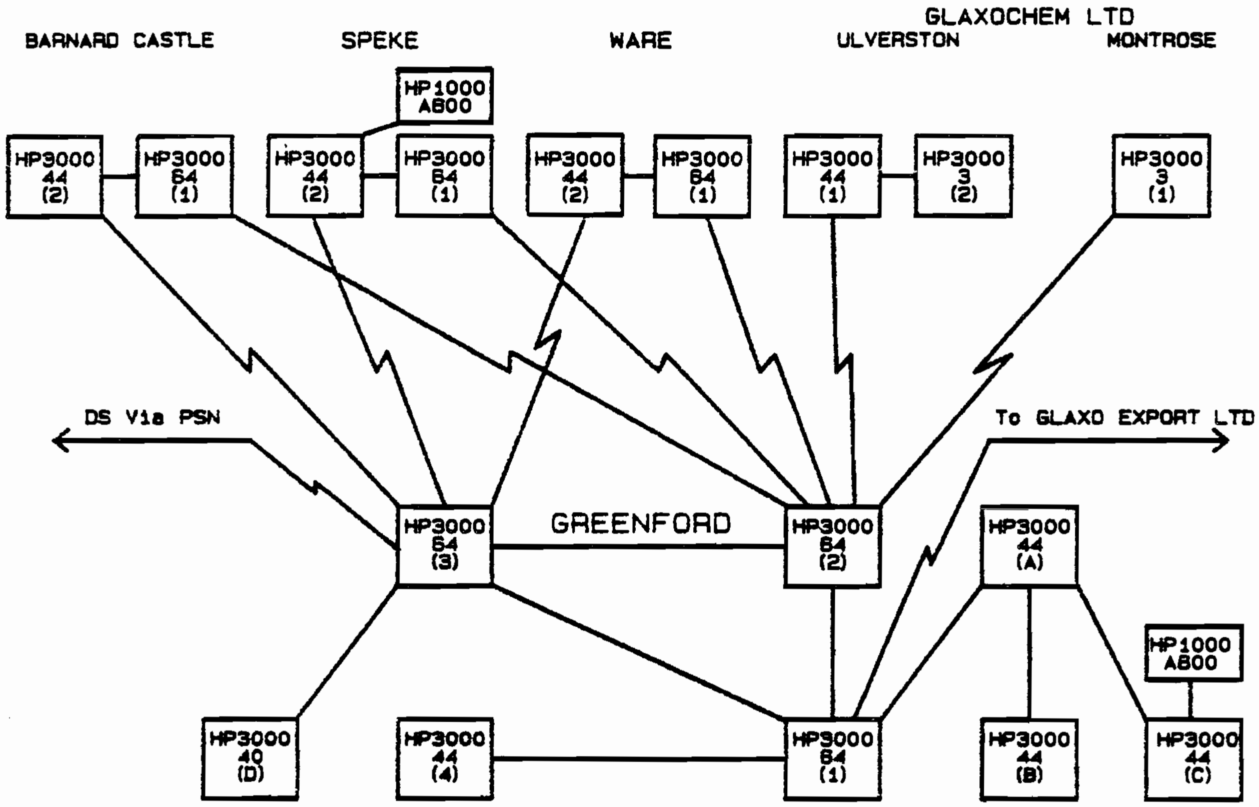
BASED ON HP3000 COMPUTERS

B. A TELECOMMUNICATIONS NETWORK

BASED ON DIGITAL EXCHANGES

GLAXO PHARMACEUTICALS LIMITED

COMPUTER NETWORK - AUGUST 1983


BARNARD CASTLE
SPEKE
WARE
GLAXOCHEM LTD
ULVERSTON
MONTROSE
HP3000 44 (2)
HP3000 64 (1)
HP3000 44 (2)
HP1000 A600
HP3000 64 (1)
HP3000 44 (2)
HP3000 64 (1)
HP3000 44 (1)
HP3000 3 (2)
HP3000 3 (1)
DS Via PSN
To GLAXO EXPORT LTD
GREENFORD
HP3000 64 (3)
HP3000 64 (2)
HP3000 44 (A)
HP1000 A600
HP3000 40 (D)
HP3000 44 (4)
HP3000 64 (1)
HP3000 44 (B)
HP3000 44 (C)

OFFICE AUTOMATION PRODUCTS

GRAPHICS PACKAGES – from May '81

MODELLING PACKAGES – from May '81

WORD125 – from Dec '81

HPMAIL – from Feb '82

HPWORD – from Jun '82

HPTELEX – from May '83

HPMAIL

A CHRONOLOGICAL SUMMARY

Feb '82	BETA TEST SITE
May '82	AGREEMENT TO PURCHASE
Jul '82	SOFTWARE TRANSPORT
Mar '83	THREE MORE COMPUTERS
May '83	AUTODIAL MODEM
Jun '83	HPDESKMANAGER ON TEST

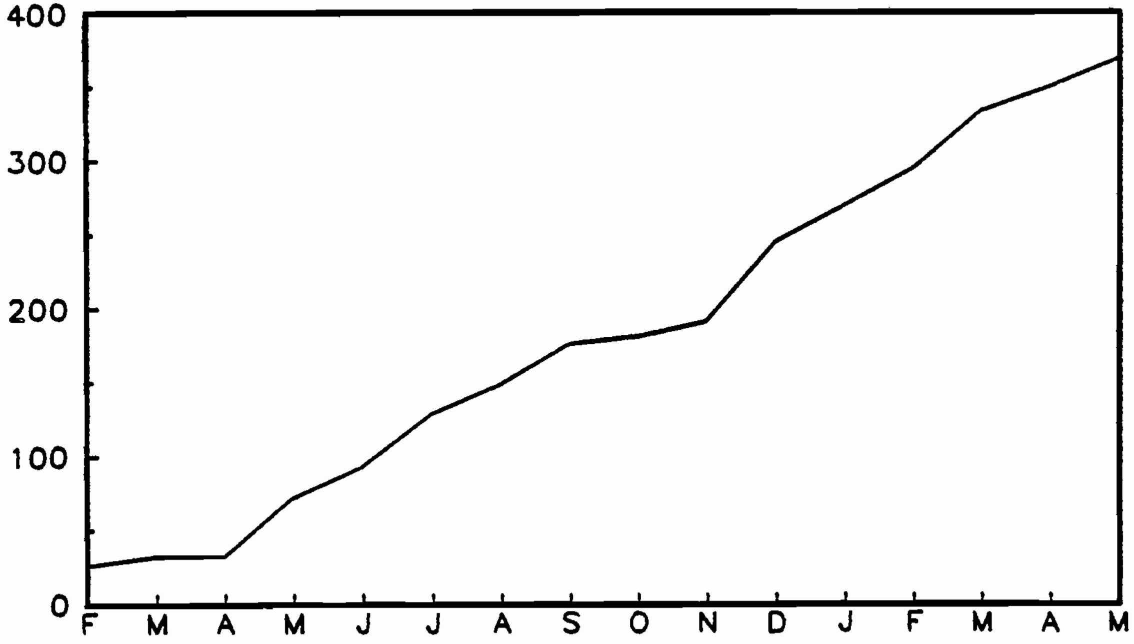
CRITERIA FOR THE SELECTION OF NEW USERS

USER GROUPS WITH THE NEED TO COMMUNICATE

READY ACCESS TO A TERMINAL

NUMBERS OF HPMAIL USERS

Feb 1982 to May 1983



HPMAIL

ITS STRONG POINTS

USER FRIENDLY

RELIABLE

ALL FILE COMPATABILITY

ACKNOWLEDGEMENTS

ANY TERMINAL

HPMAIL

ITS SHORTCOMINGS

INEFFICIENT NETWORKING

ARDUOUS HOUSEKEEPING

LIMITED MESSAGE LENGTH

DATASET LOCKING

Slide 9

WORD PROCESSING

VYDECS

WORD125

EXXON 500 SERIES

HPWORD

HPWORD
ITS GOOD POINTS

COMPATABILITY

STORAGE BY PARAGRAPHS

EASY TO USE

GOOD EDITING

NON STANDARD DAISY WHEELS

HPWORD
ITS SHORTCOMINGS

DEPENDENCE UPON COMPUTER

NON STANDARD TERMINAL

NO TRAINING COURSES

POOR FORMS ABILITY

PAGINATION DIFFICULTIES

HPWORD

HOPES FOR THE FUTURE

STAND-ALONE SYSTEM

GRAPHICS CAPABILITIES

TRAINING COURSES

FORMS CAPABILITY

IMPROVED OUTPUT

HPTELEX

ITS STRONG POINTS

EASY TO USE

LARGE DIRECTORY

ANY BLOCK MODE TERMINAL

HPTELEX

ITS SHORTCOMINGS

COMPUTER PORTS

LONG TELEXES

RANDOM DIRECTORY

POOR SOFT KEYS

ONLY BLOCK MODE TERMINAL

HPTELEX

HOPES FOR THE FUTURE

INTERFACE WITH HPMAIL

TELEX STATISTICS

ALPHABETICAL DIRECTORY

BETTER SOFT KEYS

GRAPHICS PACKAGES

AUTO & MULTIPLOT FOR 2647

DSG

EASYCHART

HPDRAW

GRAPHICS 125

OTHER SYSTEMS

FINANCIAL MODELLING

PRESTEL

EXTERNAL DATABASES

LOCAL AREA NETWORKS

Linguistic Aspects of Word Processing

by

Franz-Josef Boll

Word processing is well on the way to being fully integrated with data processing and telecommunications. But at the same time there is a tendency to forget that word processing deals with texts, with languages in written form and that modern languages can be very complicated from it's inherent structure and do not always easily fit into a machine's requirements. It is therefore essential that linguistics be considered when designing and implementing a word processing system.

"Intelligent" word processing systems will not only regard texts as a sequence of meaningless characters, but will try to recognize words, sounds within a word etc.

The main points we will discuss here are characters sets, vocabularies, frequency of word usage, automatic correction aid and automatic hyphenation.

The ideas presented here are the result of my work in designing and implementing the word procesing system **IDT** - Integrated text and data processing, where multi-language support is one of the features available to the user.

Writing Sounds

In contradiction to the hieroglyphics common to, say, Asia, written texts in European languages consist of letters representing sounds. Generally a word is made up of more than one letter, and each letter or group of letters, such as a diphthong or a consonant group like the **ch** in German, depicts a sound.

So we see, words are a combination of letters, of alphabetic symbols. The first logical step in many algorithms of word processing, therefore, is to recognise orders of characters, to be able to isolate a word from the context, to comprehend semantemes, which are frequently, but not always, terminated by spaces or punctuation marks. Before such tasks as "spelling aid" or hyphenation can be overcome, it is necessary to isolate "current" words.

The definition of national character sets, though, differs substantially. The ASCII character set defines **A:Z** and **a:z** as **alphabetic**, this is implemented in the hp 3000 instruction set at firmware level and can be used easily by SPL instructions like **MOVE WHILE AS** etc. (Note: COBOL uses a different definition!).

The set of alphabetic characters of the ASCII standard is a subset of other European characters sets. There are countless national peculiarities, for example the **ç** - c with cedilla - in French, **Ø** in Sweden, umlauts and the **ß** in German. Naturally the use of accents, diacritic marks, must also be taken into consideration, as they are part of an alphabetic word.

Complete National Character Sets

What does all that mean in a practical sense? Well, on user level the support for languages and it's characters should be a must. Many systems already fulfill that requirement, but only few of them allow you, for example, to switch from one language to another without major problems. HP's terminals now usually support 7 character sets, by the way we are missing Italian and Portuguese! But scanning, searching and upshifting must be done by application software still - printer support to a certain extent too. For example it is mandatory to replace the German ä by and **Ä** in many algorithms, it might even be useful to replace the French **à** by an **A** when searching for strings.

The word processing systems IDT uses procedures that branch into a different "case" for each character set or language, and test and upshift all not blank not alphabetic ASCII characters, where necessary. Certain bit combinatians are replaced by other ones or are recognised as part of a word.

In this context it is worth while mentioning printer output as an interesting problem. Software based solutions surely are the future, some printers, for example, have firmware ROMs for different sets, an other approach is to synthesise character sets on daisywheel printers, the OLYMPIA EUROTRONIC typewriter uses that technology and is able to print 11 different character sets without the necessity to change the print wheel! Laser printers and software controllable ink spot printers will offer additional capabilities.

Living Vocabulary

Surely one of the great fascinations of modern languages is their complexedness. Alone the vocabulary of a language rich in cultural tradition and with a long historical background is a great challenge!

Let us take the German language, for example:
The three volumes of the "small" DUDEN dictionary contain a total of approx. 75,000 keywords making it a rather limited reference book. The DUDEN for orthography contains about 200,000 entries. And these are only basic word forms! Countless other formations result from conjugation, declination, and compound words can be derived. German is especially complex in that respect.

The dictionaries again only contain a basic vocabulary, a set of words regarded as important from a certain intellectual point of view. In order to make a language "complete", many kinds of words, e.g. names, day to day expressions, technical, scientific terms must be considered, too.

Another example: German chemical science is said to have a vocabulary of 500,000 words, German medical science has about 80,000.

In addition, these vocabularies are constantly being subject to changes and extensions. Dictionaries usually drag a lot of dead weight around with them in the form of old-fashioned or obsolete expressions, whereas it is virtually impossible to find there "new" words like in German **Lebensqualität** or **Windsurfen** or **Eckdaten**.

It is now interesting to observe the frequency of word usage: Among the astronomical number of words and word formations there are only a few, that are used repeatedly. In German for example, the four most frequent words cover about 10 % of a text, and only 300 word forms cover 50 - 70 % of all lexical units in a text. These words are usually evenly distributed in all texts, whereas others - the less frequent ones - usually are not. While a certain basic vocabulary of comparatively well-used words can be found in the writings of all authors, there are other words usually used only in a certain context, by certain authors, at certain times.

Again some examples :
Many social classes have their own language, slang or jargon and words used by teenagers would hardly be found in the world of officialdom.
There are regional differences:
Within the German language. **Plaste** and **subbottern**, for example, frequently used in East Germany are unknown in the Federal Republic, the Swiss have no **ß**, what we call **Bürgersteig** is in Austria **Gehsteig**.
We all know of the differences between American and British English, particularly in the spelling.
The South American words **carro** and **estampilla** are **coche** and **sello** in Castile.
In Norway there are even two dialects each with all the characteristics of a written language: **Nynorsk** and **Bokmal**.

Correction Aid by Computer.

All these peculiarities lead to consequences for the modern wordprocessing system. First of all for the correction aid:

Each word of a text is isolated and compared with a dictionary. If it can neither be found there nor derived from a word in the dictionary, then it is marked as doubtful. For example, the word is shown inversely or the cursor positioned at the beginning of the word. This kind of correction aid can check the spelling of a word, but not the possible mistakes in capital letters. At present it is also almost impossible to make semantic analyses. **Form** and **from** could both be correct and yet at the same time, depending on the context, a typical case of letters mistakenly turned arround.

If any correction aid system is to be more than a sales gag, it must be implemented thoroughly.

The problem of the immense number of possible words can be solved in various ways:

1. through a comprehensive standard dictionary containing a large number of word forms and based on statistical analysis.
2. providing specialized dictionaries to complement the above standard edition.
3. by enabling the user to enlarge these dictionaries extensively.
4. by implementing a "learning" system within the word-processing system which automatically absorbs the vocabulary of existing texts, or where the words which the processor does not know can be entered into the dictionary by pressing a key. After all we humans can only increase our vocabulary and bring it up to date when we never stop learning.

By combing the general and specialized parts of the dictionary and adding this ability to "learn" we can also solve the dissemination problems. The learning function will always exactly reflect the vocabulary of the latest context. Almost like a particularly keen pupil, it will "note" those very words that are presented to it day in day out.

An efficient implementation demands quick and easy recognition of the frequent words, whereby the borderline between "good" and "bad" in today's data-processing mainly lies in avoiding disk accesses. Another problem is to organise direct and random access to a large mass of data, one answer being hashing. In general, access to the dictionary should be in layers, as the real vocabulary is divided into different, logical levels, too.

The correction aid could be very valuable for scientific works. In particular, because it is difficult to find a mistake in a word like **Desoxyribonukleinsäure**, it gives you a headache, too. By the way, it might not be a bad idea to keep such monstrous words in a comprehensive constant register, which would have to be tailored to the users needs, of course. This would not only mean time saved on the text input, but would also reduce the quota of mistakes to a minimum.

Automatic Hyphenation

A further vital element to modern word processing is automatic hyphenation. Quite a number of manufacturers seem to be just too lazy to make a hyphenation routine part of their software. But there is no doubt in the usefulness of such an aid when large amounts of text have to be dealt with.

What should a hyphenic algorithm actually achieve? It examines words in the text for possible hyphenation with the aim to

1. recognise all or almost all the possible hyphenations
2. keep the rate of mistakes low.
3. hyphenate all frequent words correctly and
4. work efficiently

It is certainly not child's play to design a good hyphenic algorithm, but there is really no secret to it.

The Romanic languages are comparatively easy to deal with. In Spanish for example, words could be hyphenated after every other letter in many cases, as in **a-ta-vi-e** or **es-pi-a**.

Germanic language are quite a different problem, however:

1. Prefixes and the rest of the word must be handled separately, whereby prefixes may appear in the middle of the word.
2. A similar problem arises with suffixes, especially in the English language.
3. In German as well as in Scandinavian languages compound words are often to be found, and they are often difficult for the computer to identify, e.g. **Not-aus** but **Spitz-maus**. There is practically no limit to compound words in German.
4. Words are often hyphenated depending on their meaning. The English language is particularly rich in this respect, for example

   le-gate - to leave by will
   leg-ate - a Pope's ambassador

   sec-ond - next after the first or sixtieth part of a minute
   se-cond - to transfer to a special assignment.

   In this case we recommend that these known words be left unhyphenated. It will take many years before computers are in a position to tackle texts from a semantic point of view.
5. Certain hyphenations are possible but the results are rather ugly. The word **Urinstinkt** is one of the classical examples in German, and unfortunately computers have a rather undeveloped sense of aesthetics.
6. In German there are two special cases:
   One ist the peculiarity that **ck** becomes **k-k** when separated, although never in proper names, where it cannot be parted.
   And consonants which disappeared when combing words reappear when the word is hyphenated.
   Schif(f)fahrt is an example.

In spite of the above-named difficulties, it does not pose too great a problem to write an algorithm with a low rate of errors. It is easy to achieve 95 % correct hyphenations in the German language and 98 per cent should not be impossible. In some Romanic languages a higher level is within our reach.

An Hyphenation Algorithm.

The following approach is practical for most of the languages named. The Duden dictionary gives us the information: Words should be hyphenated "according to the syllables which automatically arise when speaking slowly", which is absolutely no use at all as an instruction to the computer. Besides that it presumes one knows how to pronounce the words. Nevertheless the fact that syllables serve as a kind of phonetic unit can be the start to solving the problem. After all, what is the definition of a syllable? It must have at least one vowel and must be pronounceable. Generally it also contains at least one consonant, besides the consonants serve as a kind of bridge between the different syllables.

The principle consideration for an algorithm then is to analyse words as a sequence of vowels and consonants and use this information for our conclusions regarding the hyphenation. Therefore in the initialization step each letter will be examined if it is an element of the vowels or not. The definition of this set differs considerably. Practically all languages contain the vowels A, E, I, O, U, but German has the additional umlaut vowels and Danish contains e.g. the Å. In some languages the letter Y functions as a vowel too, depending on its position in the word.

In the main procedure the word is dealt with from left to right, after the search for the first vowel of the syllable we branch off in one of various possibilities. This depends on the case where we have a diphthong, a single consonant following the vowel or a consonant group. Consonant groups again are tested if they belong to a group of sounds, that are a unit. The "standard" rule says that the last consonant belongs to the second syllable as in **Land-gut** and **ge-hen**. However there are groups of consonants in all languages which form a phonetic unit and therefore cannot be separated. Examples of these groups are **ch** and **sch** in German, **nh** in Portugese, **ll** in Castilian and **skj** in Scandinavian.
Using this algorithm a primary word of any length can be divided by working our way through the syllables.

The next step is to deal with such structures as prefixes, suffixes and compound words. Prefixes might be investigated at the beginning of the program. However German prefixes appear in the middle of words as well as at the beginning, as in **Ver-ord-nung** and **Da-ten-ver-ar-bei-tung**. Thus this check must not only be carried out at the beginning of a word but in many cases at the beginning of each syllable, too. This way a large number of the complicated cases of hyphenation can be coped with, provided, of course, the corresponding queries are not too trivially programmed. For reasons of speed and clarity the investigation of prefixes can be embedded in a multiple branch operation according to the first letter.

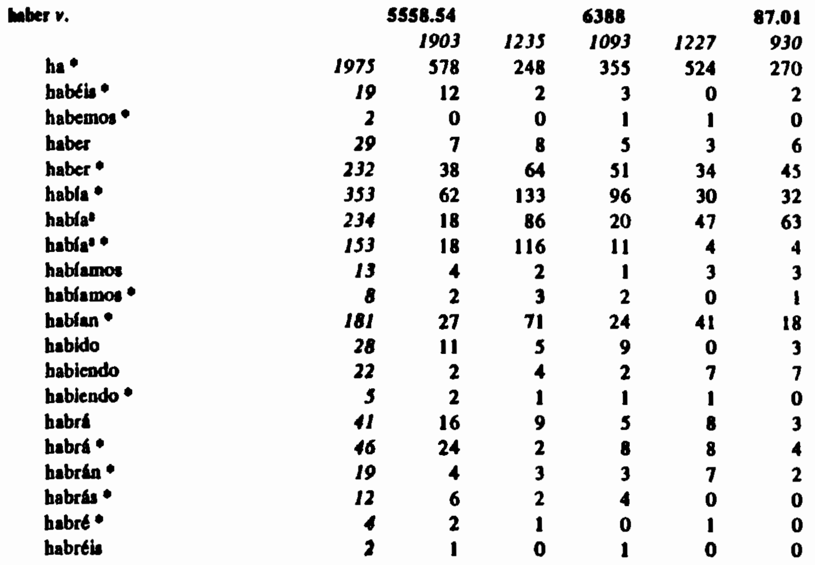
Suffixes also deserve careful consideration, e.g. -less -ment -dom in English. Suffixes present a similar problem to prefixes as they do not always appear at the end of the respective lexical unit. The greatest problem is posed by compound words, but analyses on the kind and frequency of compound words could contribute towards a solution in this case.

Foreign words, and that includes the entire technical and scientific terminology are often easier to analyse than a true German word that has been put together from other originally German words. Furthermore a good algorithm will master many cases which just seem doubtful to a secretary, while the errors the program makes strike one in the eye immediately.

Edinburgh 1983

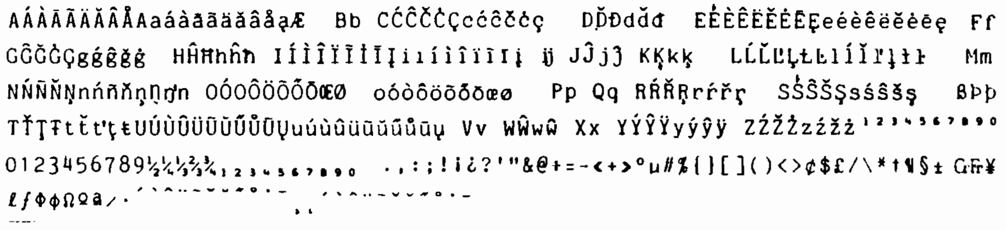
The combination of a program for hyphenating and an exeption storage possibility is absolutely essential, as it is about the only means of getting a grip on practically every hyphenation.

A final remark: A hyphenation program should be able to indentify as many hyphenations as possible and correctly. A computer set system, on the other hand, should avoid hyphenating if possible. Type set separates e.g. the German word **ei-ne** because it is allowed grammatically, but a scientist or an author with a sense of aesthetics might find hyphenation in itself as ugly, especially when the last word of a page is split or compound words are hyphenated in the middle of a word. Not to mention the possibility to separate **a-broad** in English.



haber *v.*		**5558.54**		**6388**		**87.01**
		1903	*1235*	*1093*	*1227*	*930*
ha •	*1975*	578	248	355	524	270
habéis •	*19*	12	2	3	0	2
habemos •	*2*	0	0	1	1	0
haber	*29*	7	8	5	3	6
haber •	*232*	38	64	51	34	45
había •	*353*	62	133	96	30	32
había^{s}	*234*	18	86	20	47	63
había^{s} •	*153*	18	116	11	4	4
habíamos	*13*	4	2	1	3	3
habíamos •	*8*	2	3	2	0	1
habían •	*181*	27	71	24	41	18
habido	*28*	11	5	9	0	3
habiendo	*22*	2	4	2	7	7
habiendo •	*5*	2	1	1	1	0
habrá	*41*	16	9	5	8	3
habrá •	*46*	24	2	8	8	4
habrán •	*19*	4	3	3	7	2
habrás •	*12*	6	2	4	0	0
habré •	*4*	2	1	0	1	0
habréis	*2*	1	0	1	0	0

Cutting of a Word Frequency Dictionary for Spanish



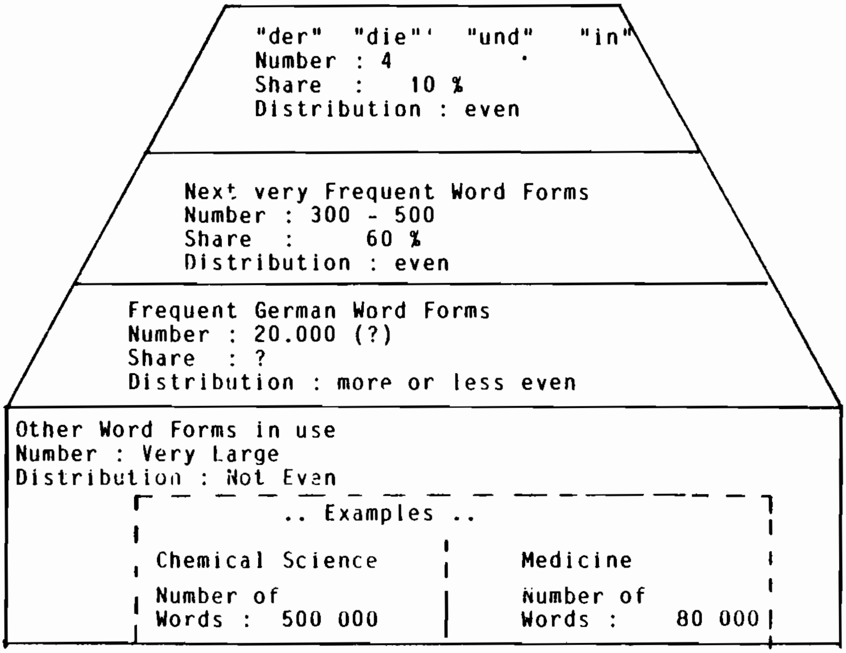
AÁÀĀÃÄĂÂÅAaáàāãäăâåąÆ Bb CĆĈČĊÇcćĉčċç DĎĐdďđ EĖÈÊËĚĖĒĘeéèêëěėēę Ff
GĞĜĠĢgǵĝğġ HĤĦhĥħ IÍÌÎÏĨİĪĮiıíìîïĩīį ĳ JĴjĵ KĶkķ LĹĽĿĻŁĿlĺľŀļłƚ Mm
NŃÑŇŅnńñňņŊŋn OÓÒÔÖÕŐŌŒØ oóòôöõőōœø Pp Qq RŔŘŖrŕřŗ SŚŜŠŞsśŝšş ßÞþ
TŤŢŦtťt'ţŧUÚÙÛÜŨŬŰŮŪŲuúùûüũŭűůūų Vv WŴwŵ Xx YÝŶŸyýŷÿ ZŹŽŻzźžż¹²³⁴⁵⁶⁷⁸⁹⁰
0123456789½¼⅓⅔¾₁₂₃₄₅₆₇₈₉₀ .,:;!¡¿?'"&@+=-<+>°µ#%{}[]()<>¢$£/*†¶§± G₣¥
ℓƒΦφΩ℧a⁄· ´`^¨~˘ˇ˝˚˙¯ ´`^¨~˘ˇ˝˚˙_

Elements of European National Character Sets

FIRST FIVE HUNDRED

	USAGE		FREQUENCY		DISPERSION	
1	de *p.*	30526.45	de *p.*	35144	lumbrera *n.*	100.00
2	el *a.*	25100.98	el *a.*	28440	misticismo *n.*	100.00
3	la *a.*	21320.23	la *a.*	23937	pesquisa *n.*	100.00
4	y *c.*	14049.55	y *c.*	14946	vestíbulo *n.*	100.00
5	a *p.*	13196.45	a *p.*	13720	fin *n.*	96.55
6	en *p.*	11602.37	en *p.*	12922	el *pn.*	96.19
7	él *pn.*	11104.34	él *pn.*	11543	a *p.*	96.18
8	que *pn.*	8752.35	que *pn.*	10167	sin *p.*	95.25
9	ser *v.*	8692.76	ser *v.*	9907	tanto *aj.*	95.23
10	que *c.*	6087.81	que *c.*	7518	mejor *aj.*	94.20
11	haber *v.*	5558.54	no *av.*	6900	hasta *p.*	94.18
12	su *aj.*	4854.90	haber *v.*	6388	ánimo *n.*	94.07
13	no *av.*	4688.53	yo *pn.*	5953	y *c.*	94.00
14	un *a.*	4497.30	su *aj.*	5596	cuenta *n.*	93.83
15	por *p.*	4397.60	un *a.*	5033	labrador *n.*	93.75
16	con *p.*	4295.25	por *p.*	4700	fiel *aj.*	93.68
17	una *a.*	3387.94	con *p.*	4667	ninguno *aj.*	93.57
18	yo *pn.*	3110.65	una *a.*	3708	por *p.*	93.57
19	para *c.*	2470.22	ella *pn.*	3043	dónde *av.*	93.52
20	ella *pn.*	2425.91	para *c.*	2682	quien (relat.) *pn.*	93.36
21	este *aj.*	2225.19	este *aj.*	2651	más *av.*	93.31
22	más *av.*	2204.94	estar *v.*	2501	alguno *pn.*	93.28
23	lo *a.*	2109.27	lo *a.*	2387	capaz *n.*	92.94
24	como *c.*	2001.97	tener *v.*	2384	delicado *aj.*	92.80
25	estar *v.*	1929.54	más *av.*	2363	pesado *aj.*	92.80
26	tener *v.*	1920.00	ello *pn.*	2292	pretender *v.*	92.70
27	todo *aj.*	1772.78	como *c.*	2265	poder *v.*	92.32
28	hacer *v.*	1585.94	ir *v.*	2064	llegar *v.*	92.27
29	poder *v.*	1541.70	decir *v.*	2037	crear *v.*	92.16
30	decir *v.*	1488.46	todo *aj.*	1951	para *c.*	92.10
31	ello *pn.*	1477.92	tú *pn.*	1928	con *p.*	92.03
32	ir *v.*	1345.75	hacer *v.*	1868	palabra *n.*	91.83
33	pero *c.*	1326.08	pero *c.*	1792	desde *p.*	91.72
34	o *c.*	1110.90	poder *v.*	1670	año *n.*	91.69
35	otro *aj.*	1063.51	usted *pn.*	1574	tan *av.*	91.66
36	sin *p.*	1002.99	o *c.*	1503	encontrar *v.*	91.52
37	dar *v.*	976.31	si *c.*	1327	frase *n.*	91.45
38	si *c.*	954.93	ya *av.*	1313	recordar *v.*	91.41
39	ver *v.*	953.84	ver *v.*	1302	una *a.*	91.37
40	ya *av.*	926.88	otro *aj.*	1255	mucho *aj.*	91.24
41	dos *nu.*	858.33	mi *aj.*	1247	cualquiera *pn.*	91.22
42	ese *aj.*	761.88	dar *v.*	1127	familia *n.*	91.20

Spanish Words sorted by Frequency



"der" "die" "und" "in"
Number : 4
Share : 10 %
Distribution : even

Next very Frequent Word Forms
Number : 300 - 500
Share : 60 %
Distribution : even

Frequent German Word Forms
Number : 20.000 (?)
Share : ?
Distribution : more or less even

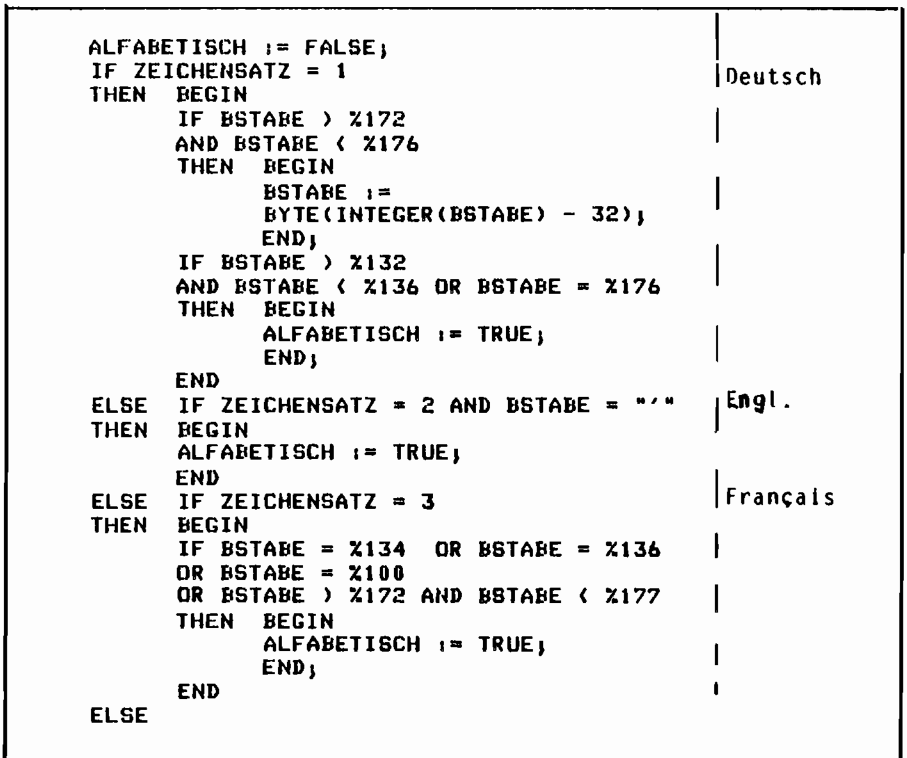
Other Word Forms in use
Number : Very Large
Distribution : Not Even

.. Examples ..

Chemical Science
Number of Words : 500 000

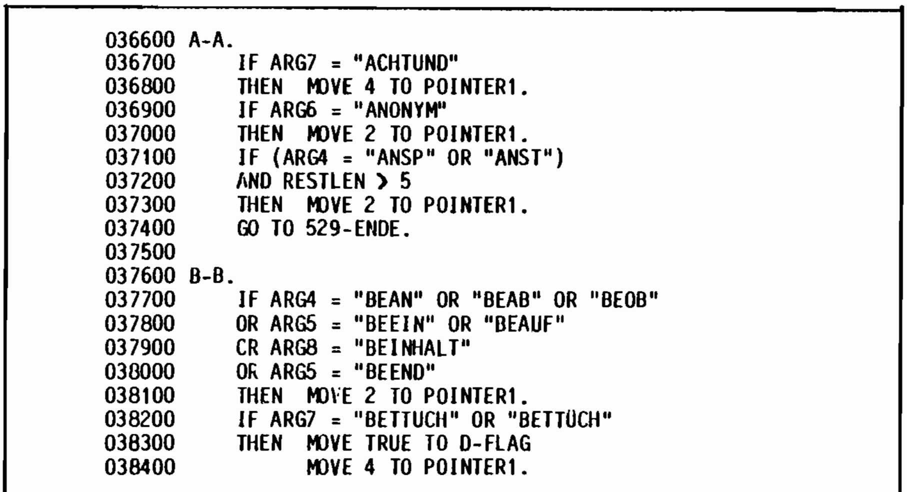
Medicine
Number of Words : 80 000

Elements of the German Vocabulary



```
ALFABETISCH := FALSE;                                      |
IF ZEICHENSATZ = 1                                         |Deutsch
THEN  BEGIN
      IF BSTABE > %172                                     |
      AND BSTABE < %176
      THEN  BEGIN
            BSTABE :=                                      |
            BYTE(INTEGER(BSTABE) - 32);
            END;
      IF BSTABE > %132                                     |
      AND BSTABE < %136 OR BSTABE = %176
      THEN  BEGIN
            ALFABETISCH := TRUE;                           |
            END;
      END
ELSE  IF ZEICHENSATZ = 2 AND BSTABE = "'"                  |Engl.
THEN  BEGIN
      ALFABETISCH := TRUE;
      END
ELSE  IF ZEICHENSATZ = 3                                   |Français
THEN  BEGIN
      IF BSTABE = %134  OR BSTABE = %136                   |
      OR BSTABE = %100
      OR BSTABE > %172 AND BSTABE < %177                   |
      THEN  BEGIN
            ALFABETISCH := TRUE;                           |
            END;
      END                                                  |
ELSE
```

SPL Procedure to Upshift and Test for non-ASCII Alphabetic Characters



```
036600 A-A.
036700     IF ARG7 = "ACHTUND"
036800     THEN  MOVE 4 TO POINTER1.
036900     IF ARG6 = "ANONYM"
037000     THEN  MOVE 2 TO POINTER1.
037100     IF (ARG4 = "ANSP" OR "ANST")
037200     AND RESTLEN > 5
037300     THEN  MOVE 2 TO POINTER1.
037400     GO TO 529-ENDE.
037500
037600 B-B.
037700     IF ARG4 = "BEAN" OR "BEAB" OR "BEOB"
037800     OR ARG5 = "BEEIN" OR "BEAUF"
037900     CR ARG8 = "BEINHALT"
038000     OR ARG5 = "BEEND"
038100     THEN  MOVE 2 TO POINTER1.
038200     IF ARG7 = "BETTUCH" OR "BETTÜCH"
038300     THEN  MOVE TRUE TO D-FLAG
038400           MOVE 4 TO POINTER1.
```

COBOL Procedure to recognize Word Compositions for a German phenation Program

Data Model
Financial Modelling for the HP3000

by Edward Humphrey
Assyst-Raet Group

1. General Concepts of Financial Modelling

Company planners, line managers, accountants, internal consultancy staff are a selection of the people who produce financial analyses of various kinds. General Ledger systems deal with historical data going back over months and sometimes years. In contrast Financial Modelling systems look at the future.

There is some overlap between these two application areas. General Ledgers are often used to predict cash flow based on budgets or forecasts and outstanding balances in debtors and creditors accounts. Financial Modelling systems may use historical data as a starting point for future projections. Historical data may also be used to evaluate alternative strategies by manipulating past results to produce sensitivity analyses of business activity.

Neverthless the activity of modelling is restricted to the future. It is not a record keeping activity. The users of financial modelling referred to above will typically develop applications for

:- budgeting
forecasting
longrange planning
sensitivity analysis

The common thread in these applications is time. These activities take place around a spreadsheet which is simply a piece of analysis paper. The spreadsheet normally shows time on the horizontal axis and data items on the vertical axis. There are exceptions to this but most financial modelling systems will deal with such situations.

The problem with performing modelling tasks on analysis paper is that invariably the analysis must be repeated, using different assumptions. Wise planners do their analysis in pencil, but the paper very soon wears out.

There is always an element of uncertainty because we are dealing with the future. With any mathematical, econometric or financial model certain assumptions are made about future events. It is variations in these events which cause the repetition of the analysis, and waste time for the person performing the analysis.

This is where computerised financial modelling systems are of benefit. Time is the key factor for all hard - pressed executives. Financial modelling systems save time by automating the mundane calculations which make up such models for example:-

Income - Variable Cost - Fixed Cost - Overhead = Profit. over 12 time periods per year for five years is a very lengthy recalculation.

If we change the assumptions about the price structure of the product we are dealing with in the above example, then every element in the analysis must be changed.

- At a lower price more units will be sold
- Variable costs will change due to longer production runs (less set-up time).
- Fixed costs may change because we need more equipment.
- Changes to overheads will also be needed.

The principal benefits which computerised financial modelling bring to this type of analysis are therefore :-

Speed	- Immediate recalculation of part or the whole of an analysis.
Accuracy	- No recalculation due to minor errors.
Flexibity	- Easy addition or removal of information.
Effectiveness	- Expensive people can get on with productive work.
Decisions	- Are made easier because sensitivity analysis and rate of return calculations require little effort.

The remainder of this presentation will show how Data Model addresses these problems and allows end-users to become more effective.

2. Data Model Overview

This overview will show how Data Model addresses the problems outlined in the introduction. The spreasdheet described in that section is central to the operation of the system, exactly as in manual analysis. This means that users can relate to the system.

The spreadsheet is of course stored on disc. Data Model's strengths result from the separation of the activities which produce the report from the spreadsheet itself. This modular approach gives users considerable flexibility and interchangeability in modelling.

The components of the system are as follows:-

Row Definitions

A Data Dictionary (in a very restricted sense of that term) which defines what data items make up the model, how they will be described on reports, and rules for totalling.

Model Definitions

Allow the user to input data, interactively or in batch mode. Interfaces to external IMAGE, KSAM and MPE files are also provided.

Manipulation of data takes place via modelling instructions for basic calculations, financial routines (eg depreciation), and data manipulation.

Report Definitions

Vertical and horizontal dimensions of a report are defined separately for maximum flexibility.

The Row Definitions and Model Definitions combine to form the spreadsheet file on disc referred to earlier.

Spreadsheet operations may then link directly into a report, or into another model for further data manipulation and analysis.

Data Model's modularity is not restricted to the division of the functions within the system into separate tasks for set up purposes.

Model definitions may be linked directly to other model definitions, or to report definitions. Similarly, report definitions may be linked to other report definitions, or to model definitions. This approach gives total flexibility in model development. The user benefits by being able to break problems down into components, and by re-using similar earlier models on current problems.

Data Model allows many different reports to be produced from a single spreadsheet. One advantage of paper spreadsheets is the ability to stick them together to produce a large analysis. Data Model simulates this by providing for multiple reports (linked together if necessary). This allows the inherent restriction of printer or screen width which is common to many computer systems to be overcome.

All the above functions are achieved by just four standard menus. This is a considerable aid to user friendliness, since the menus are rapidly assimilated by the average user.

Ease of operation is carried through to the design of all responses. Users find it easy to get started with Data Model by using a subset of the modelling functions in the system. They can then gradually use more advanced functions and techniques to reach the pinnacle of modelling achievement.

In overall terms Data Model has far more power than simple spreadsheet processors like V*S*C**C and its many variants, in fact as with many minicomputer systems it approaches the large mainframe packages in its power and sophistication - assuming your users can reach the pinnacle of modelling achievement, as outlined previously.

3. Advanced Routines

In 1970 - 71 I worked as a student trainee in the project evaluation department of an automotive equipment manufacturer. Our standard method of evaluating projects was Discounted Cash Flow, incorporating both Net Present Value (easily calculated) and Yield methods (not so easy). Manual yield calculation is an iterative process where successive guesstimates of the return on investment eventually provide an actual yield value. The engineering directorate of the company posessed a desk-top computer, complete with magnetic stripe cards for program storage, (state-of-the-art technology). The manufacturer of this equipment will remain anonymous to avoid embarasment.

When yield calculations were required, there was a substantial saving of time in walking to the engineering directorate (5 minutes), feeding magnetic stripe program and data into the desk top (5 minutes), walking back to the finance directorate (5minutes). I should add here that the time saving was not due to innumeracy or the lack of (electro-mechanical) calculators to aid the yield calculation.

The point of this story is that hardware and software have progressed considerably in the last 12 years. Yield and Net Present value calculations are trivial to contemporary financial modelling systems. They are viewed however as advanced routines by users who have no more than an electronic calculator and a sheet of analysis paper at their disposal.

My remarks on advanced routines will be therefore be divided into two parts. The first is advanced routines which in terms of generally available software are an advance for the novice user of automated modelling. The second is advanced routines which will surprise the experienced user.

Advanced Routines - Part 1

- Amortisation of loans
- Depreciation calculations (single/multiple assets).
- Goal - seeking (Level of production required to reach a profit target).
- Discounted cash flow.

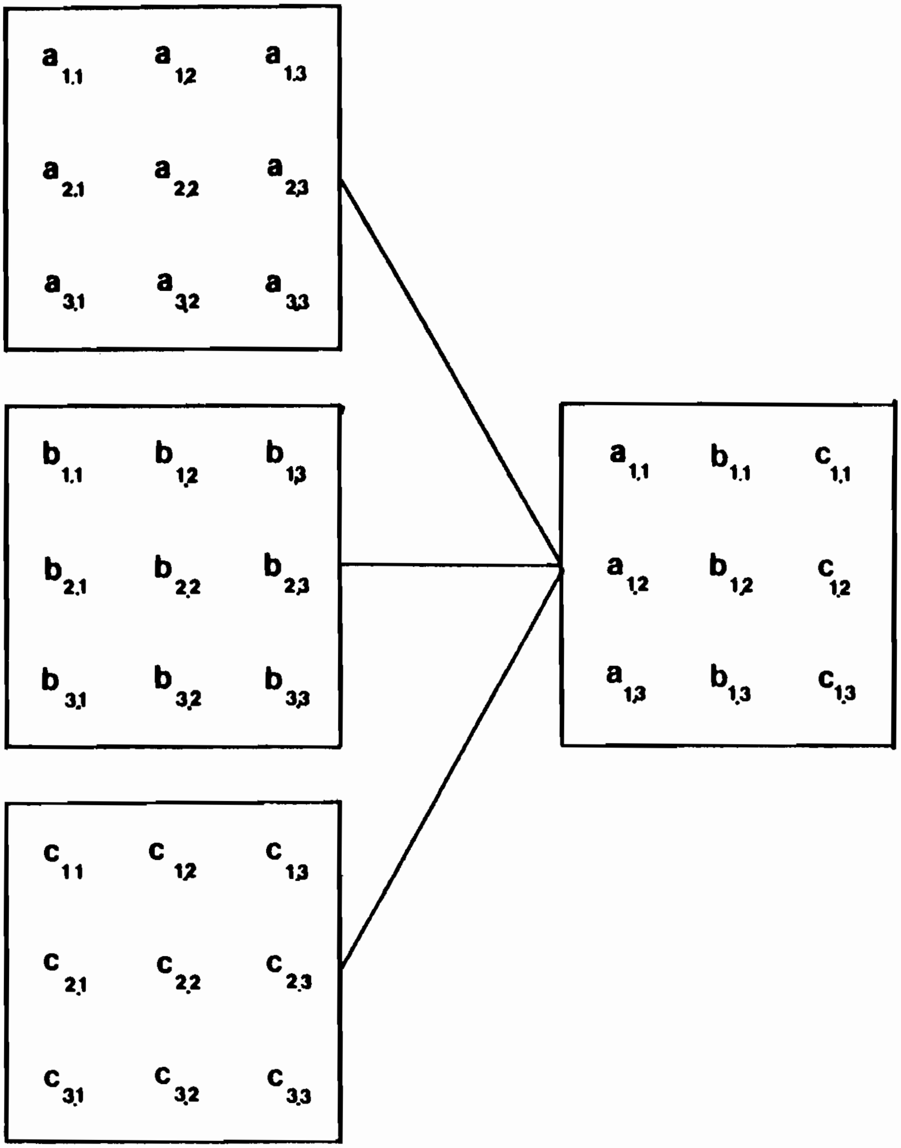
Advanced Routines - Part 2

- "JUMP" command
  This is effectively a GOTO statement. (Aplolgies to modular programming enthusiasts).

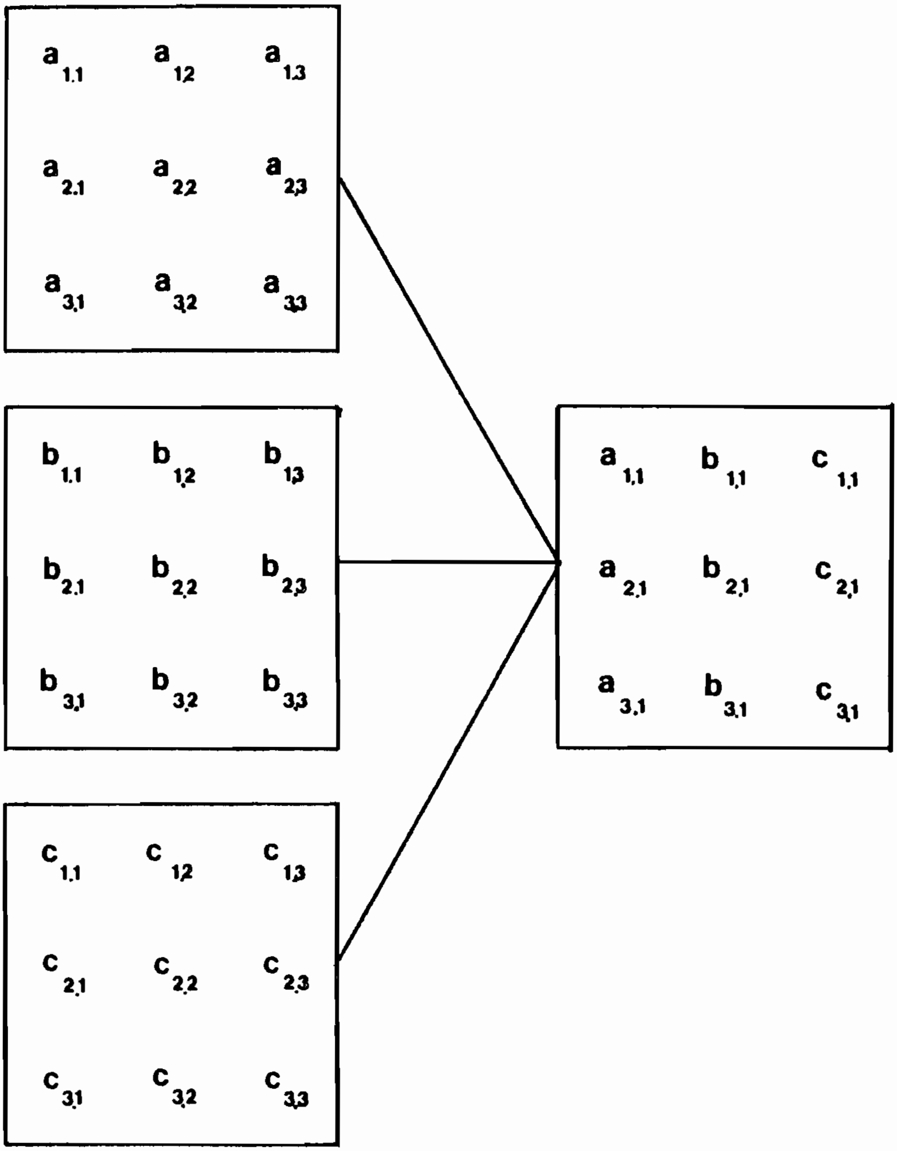
- "JUMP ON CUE" command
  This command gives the ability to PERFORM other model definitions, or parts of the current model definition.

- "INVERT" command
  Inversion allows the movement of rows and columns from single or multiple spreadsheet. The types of inversion handled are illustrated in the following diagrams.

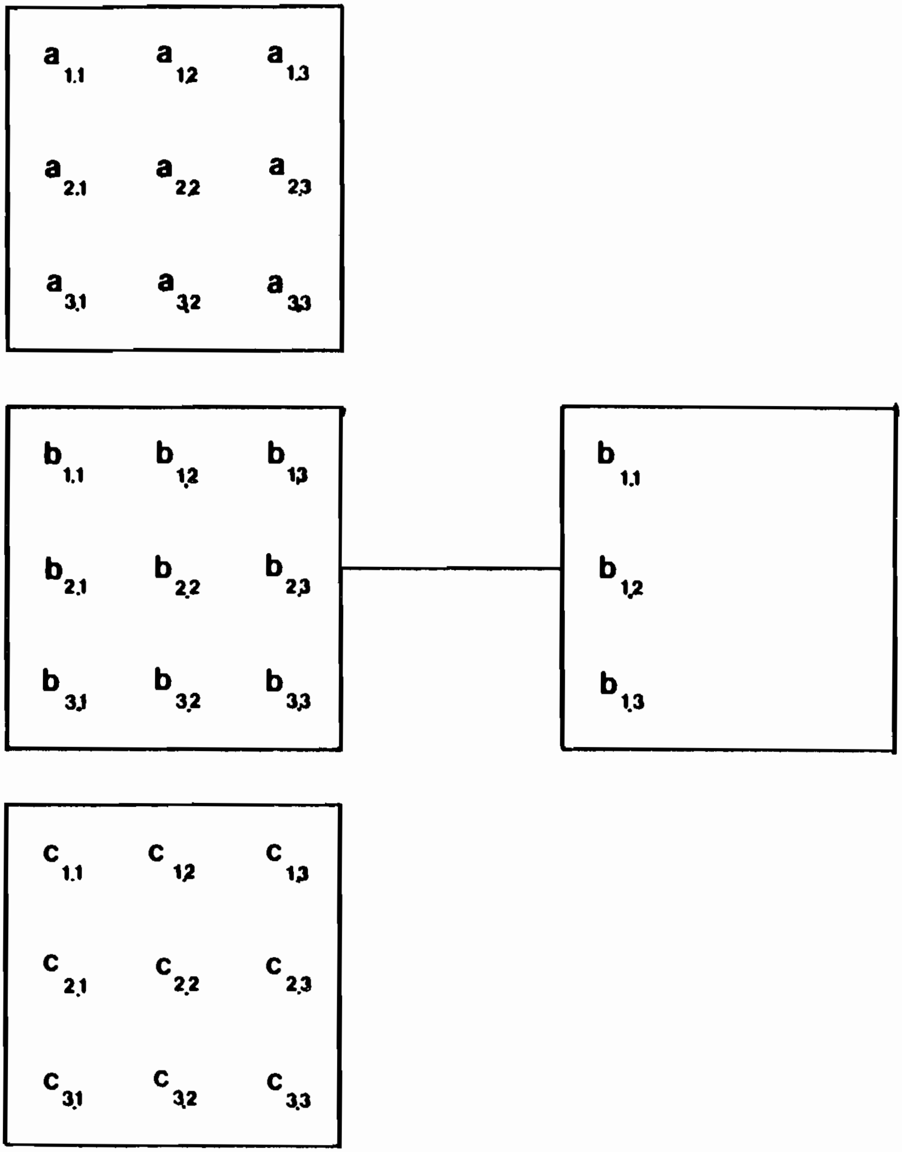
Single Row Fold (Row to column)


$a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$c_{1,1}$ $c_{1,2}$ $c_{1,3}$
$c_{2,1}$ $c_{2,2}$ $c_{2,3}$
$c_{3,1}$ $c_{3,2}$ $c_{3,3}$
$a_{1,1}$ $b_{1,1}$ $c_{1,1}$
$a_{1,2}$ $b_{1,2}$ $c_{1,2}$
$a_{1,3}$ $b_{1,3}$ $c_{1,3}$

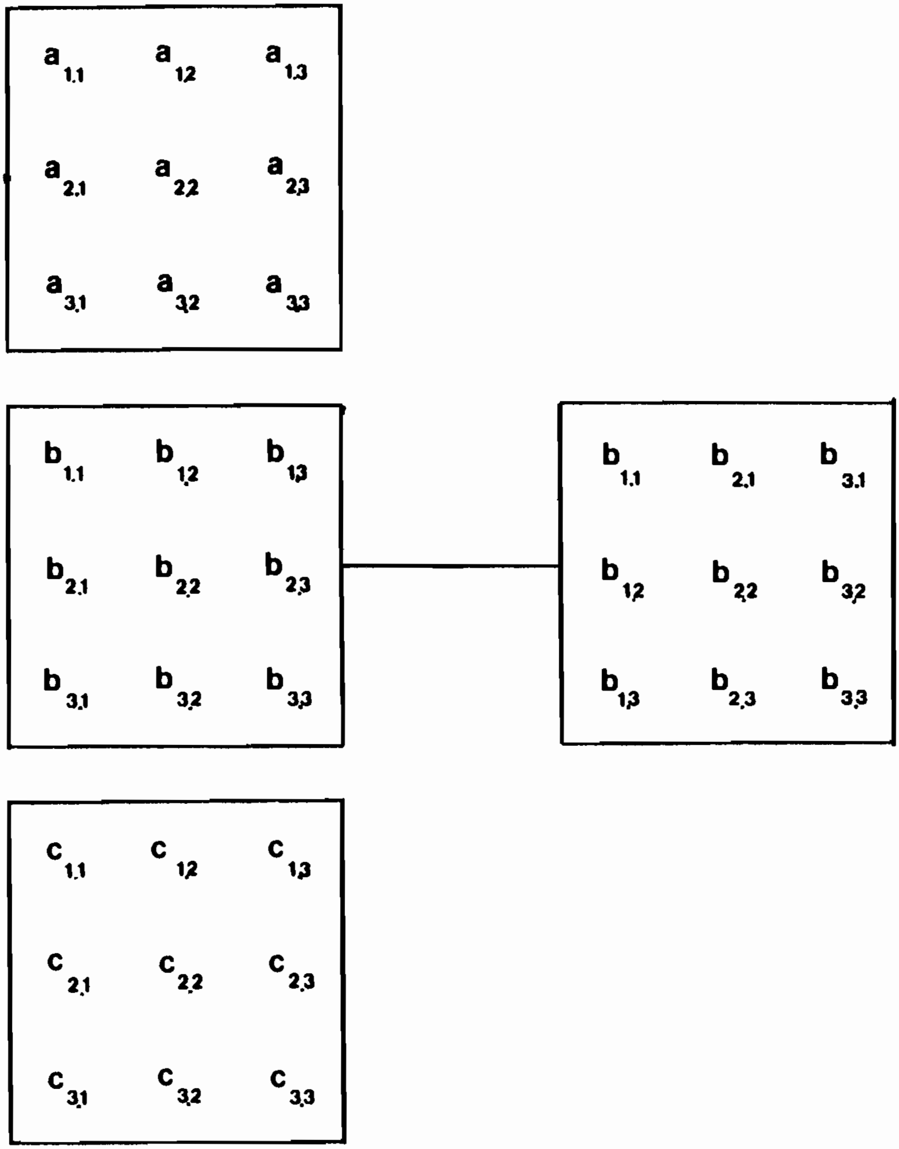
Multiple Row Fold (column to column)


$a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$c_{1,1}$ $c_{1,2}$ $c_{1,3}$
$c_{2,1}$ $c_{2,2}$ $c_{2,3}$
$c_{3,1}$ $c_{3,2}$ $c_{3,3}$
$a_{1,1}$ $b_{1,1}$ $c_{1,1}$
$a_{2,1}$ $b_{2,1}$ $c_{2,1}$
$a_{3,1}$ $b_{3,1}$ $c_{3,1}$

Single Row Inversion (Row to column)


$a_{1,1}$ $a_{1,2}$ $a_{1,3}$
$a_{2,1}$ $a_{2,2}$ $a_{2,3}$
$a_{3,1}$ $a_{3,2}$ $a_{3,3}$
$b_{1,1}$ $b_{1,2}$ $b_{1,3}$
$b_{2,1}$ $b_{2,2}$ $b_{2,3}$
$b_{3,1}$ $b_{3,2}$ $b_{3,3}$
$b_{1,1}$
$b_{1,2}$
$b_{1,3}$
$c_{1,1}$ $c_{1,2}$ $c_{1,3}$
$c_{2,1}$ $c_{2,2}$ $c_{2,3}$
$c_{3,1}$ $c_{3,2}$ $c_{3,3}$

Multiple Row Inversion (Row to column)



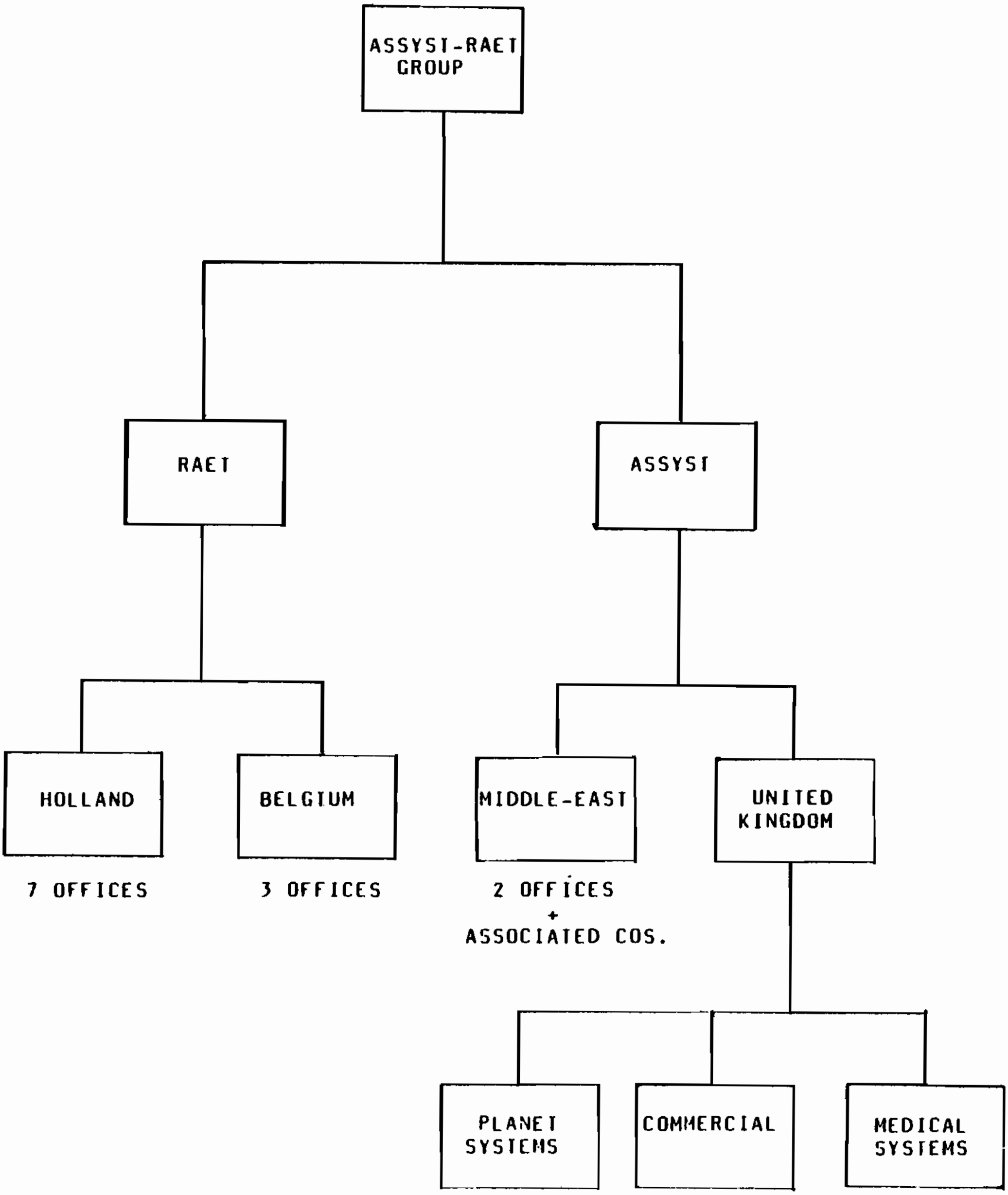
$$\begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix}$$

$$\begin{bmatrix} b_{1,1} & b_{1,2} & b_{1,3} \\ b_{2,1} & b_{2,2} & b_{2,3} \\ b_{3,1} & b_{3,2} & b_{3,3} \end{bmatrix} \rule[0.5ex]{3em}{0.4pt} \begin{bmatrix} b_{1,1} & b_{2,1} & b_{3,1} \\ b_{1,2} & b_{2,2} & b_{3,2} \\ b_{1,3} & b_{2,3} & b_{3,3} \end{bmatrix}$$

$$\begin{bmatrix} c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix}$$

"CONSOLIDATE" command

For modelling purposes this command allows the generation of reports according to the following type of structure


ASSYST-RAET GROUP
RAET
ASSYST
HOLLAND
BELGIUM
MIDDLE-EAST
UNITED KINGDOM
7 OFFICES
3 OFFICES
2 OFFICES + ASSOCIATED COS.
PLANET SYSTEMS
COMMERCIAL
MEDICAL SYSTEMS

This is achieved by adding spreadsheets (matrices of values) together. Obviously there are certain restrictions on the contents of the matrices added, or the results will be meaningless. Consolidation may be on a full spreasheet or row by row basis.

Other operators allowed in this process are

Move - copy data direct from A to B
- - Subtract A from B
* - Multiply A x B
/ - Divide A ÷ B

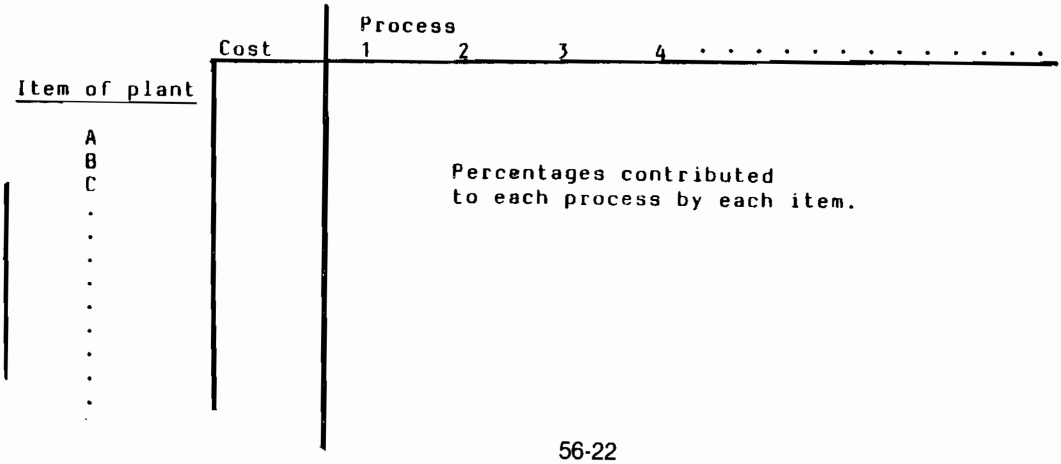
We can therefore move from simple financial models to complex statistical, econometric, and operational research applications by means of these advanced routines.

The structure of the system is such that a user at any level needs to know only those commands which are relevant to the job in hand

4. CASE Studies

Value Analysis - Sugar Refinery

A standard technique in value engineering is the allocation of plant costs to processes within a new facility. The objectives of this exercise are to determine where value is used and to plan alternative strategies for providing facilities. We were presented with an analysis of this nature for a new sugar refinery to be built overseas. The problem was that whenever the cost of an item of plant was changed, or a new item was substituted or added, the whole spreadsheet required recalculation. The analysis was not as straightforward as most since time was not included as a factor. The dimensions of the analysis were as follows:-



The objective was to multiply the cost in column 1 by the percentages in subsequent columns, with totals at the foot of each column, to determine the capital cost of carrying out each process in the new refinery.

Result - better strategic decisions.

Budgets - Drink manufacturer

This was a head - office application where the central finance function required to merge the planned budgets of its many subsidiary companies to produce a total company budget.

With the consolidation facilities available in Data Model this was a simple exercise in modelling terms, although somewhat complex in implementation due to the diverse nature of the group companies, The steps taken to achieve group, division and individual company budgets were as follows:-

1. Define a master spreadsheet, containing a row for every item of cost and income across all companies.
2. Define company, division, and group stuctures, Enter these structures as "command models", to control budget processing.
3. Enter budget data at detailed level by company.
4. Generate budget statements.

Steps 3 and 4 of this process were of course repeated ,many times until the results were acceptable to management. Note that steps 1 and 2 involve the creation of a model of the group of companies, and that this model may then be used for the reporting of actual values in comparison with budgets, evaluating the effects of results on overall cash flow, and so on.

5. Conclusion

I hope I have shown with the topics covered in this presentation how financial modelling fits into the spectrum of financial control activities, and how Data Model provides a solution in such applications. I hope I have also demonstrated the way in which advanced modelling systems go beyond the limitations of micro-based systems and provide new capabilities into which the user can develop.

In the case of Data Model, the key to these capabilities is its modularity. It allowa the same approach to financial and other modelling as we apply to programming by enforcing a modular approach in that activity.

DATA * MODEL is the registered trademark of MMI,Seattle, USA

HEWLETT
PACKARD

INFORMATION NETWORKS DIVISION · 19420 Homestead Road, Cupertino, California 95014, Telephone 408 725-8111

FROM: Bruce Woolpert
DATE: June 10, 1983

TO: Jo Anne Cohn
SUBJECT: International Users Group Conference -- Paper Abstract

Author: Bruce Woolpert
Marketing Manager
Personal Software Division

Title: Personal Computers in Networked Systems

Abstract:

The advancing power of personal computers, expanding electronic mail networks -- public and private, increasing demand for management access to "host" shared decision support data, and the growing incentives for electronic filing are all important trends having significant impact on future computing resource planning. The presentation will describe these trends and their importance, and suggest a view of the future for networked personal computer systems. The goal is to increase the productivity of business professionals, managers, and support personnel.

BWW:mr

When it comes to information technology, it has never been easy to predict the future with great accuracy. New technologies and invention often come from the least expected places. Old technologies, such as voice telecommunications, take on whole new roles in digital computer applications as VLSI and messaging transmission technologies push "old technologies" forward. If one prefers a quiet deserted island, the information processing industry will probably never provide a happy employment home.

Technology changes and advances in the next few years are likely to have an even greater impact on what's expected from our computer resources. They will also dramatically touch the lives of our user communities — (involvement in the direct job activities) of our professional, secretarial, clerical employees and our bosses.

What I'm referring to is the impact that the personal computer will have on almost every facet of our business. It is often said that the 1960's was the decade of the mainframe, the 1970's the decade of the minicomputer, and the 1980's the PC decade. Market projections show that minicomputer and mainframe portions will comprise only 40% of total computer market revenues by the late 1980's. The personal computer will have the largest and fastest growing portion at 60%. These projections tend to make most computer practitioners abit uneasy. In many companies today, there is no plan for the purchase or integration of P.C.'s into the company's information systems. Problems with data incompatibility, with document interchange incompatibility, and with inconsistent user interfaces that make cost effective training impossible — all are problems requiring MIS department concern.

Yet the advantages of today's PC's are significant enough to ever increase PC growth and overshadow the problems. The range of software available (from cattle feed scheduling, word processing, to home entertainment), prices that challenge today's terminal prices, and a much more responsive user interface all weigh heavily in higher PC growth. These are all of significant benefit to the large user communities within our companies.

We couldn't afford to move our companies into the 1980's without addressing the productivity of our professional employees. Many companies have already calculated that lower costs (in order to remain competitive) will necessitate a planned significant improvement in "white collar" professional productivity. In the U.S., this employee group comprises 54% of all workers and generates 75% of all payroll costs. A major U.S. manufacturing company has set goals for productivity advancement as part of its financial planning process — a 15% increase in professional productivity within 2 years and 50% within 5 years. The productivity tools that they look to deliver these increases are office automation and personal computing.

Today we find about one PC for every 13 office workers. These machines are shared among many users who all require access to "PC software favorites" like Visicalc, Wordstar, and MultiPlan.

What will happen to increase the PC population from 1 PC for every 13 users to 1 PC for every 2 workers by the late 1980's — and 1 PC for every 6 workers by 1985? Eventually, we will find a PC on every desk.

Will it be more applications software, greater prestige appeal of having one on your desk, or some other reason? I believe that the impetus behind this growth will come from a whole new population of PC users and a closer integration with a company's information resources. It will come from the merging of the terminal and PC into a single machine — I call them networked workstations — coupled with a whole new set of information management, information access, decision support, and communications functions.

A networked workstation will probably come in several packages or "flavors" for a range of PC user communities. For example, a sales organization would need a lightweight, battery-powered portable machine while a design engineer would require a large display machine for CAD applications. All of the members of the workstation family would need to meet certain common needs and provide data compatibility.

All users will require the common core office productivity tools: word processing, spreadsheet/decision support, data base, graphics, and minicomputer and other network datacommunications.

There are four major benefits to be gained through tying workstations into our "host" computer networks. First, electronic mail (and voice mail in the future) addresses the need for more efficient and lower cost communications. It dramatically reduces the amount of lost time spent on telephone tag. With the introduction of new "electronic mail public networks", each workstation can send messages to individuals in other companies at a fraction of the cost of postage and it arrives in minutes instead of days.

The second benefit of networking is information access. Today, a business spends about 35% of its DP budget on entering and maintaining data. Yet too little of this information is used by management to actually manage the business — conduct analysis of problems, ad hoc reporting, and future planning. To make this readily available and usable by management, additional work must be done to improve our information access tools. They need to be easier to use but also more powerful in accessing a wider range of data. The local processing power of the workstation will help here. Progress is being made in data base management and user interface software to take advantage of each company's "information resource." In addition, advances in disc storage technology will make it possible to cost effectively have years vs. months of data available on-line.

Thirdly archival storage also greatly benefits from reductions in data storage costs. In our offices, we expend a great deal of labor and floor space to hand file and retrieve information. Although, the age of electronic filing isn't upon us as yet, the economics of computer archival storage will change in the next several years through advances in disc storage technologies. When one includes the cost of labor and floor space to manually file and retrieve a document, the filing cabinet looks like it will become "extinct" in the office of the 1980's. The workstation will require transparent access to shared storage devices to access these electronic files.

Lastly, workstation shared peripherals also include laser printers and color drafting plotters. While these devices are largely located only in the computer room environment today, advances in hardware design will put lower cost devices in the work areas of our offices in the future. In work groups or departments, they can be shared by several networked workstations.

Because technology trends will continue to make such shared services (data, print, mail, etc.) even more attractive in the future, we can expect to see a much closer coupling of workstations with host minicomputers, mainframes and networks in the future. The sharing of important information through shared disc storage and electronic mail transmission will greatly enhance the effective use of the core productivity tools running locally on workstations.

Access to shared printers and plotters will expand the capability of producing high quality output from each workstation without the necessity of purchasing an expensive peripheral device for each workstation.

I hope that this paper has suggested some useful ideas on the future of the personal computer within each of our business organizations. These new technologies may be coming at the right time to help address our needs for increased productivity of the wide range of professional and support users in our organizations.

VIEWDATA CHALLENGE TO TRADITIONAL DATA PROCESSING

by David C. Bennett, U.K. Sales Manager,
Air Call Computer Systems Limited

Computer Museum

Videotex Systems have been one of the most talked about but least understood developments in our industry for a decade. Because of their attractiveness and ease of operation, they have been the subject of excessive marketing 'hype' and overzealousness by starry-eyed Salesmen. More often than not, they have convinced companies to install videotex systems without any idea where, or even if, any benefits exist. This usually leaves data processing departments to "carry the can", involving an after-the-event evaluation which, our research has discovered, shows up serious shortcomings in the earlier systems which almost disqualify them as contenders on a serious basis for effective business solutions.

Additionally, like the computer industry, Videotex Systems have acquired a terminology all of their own (Teletex, Viewdata, Gateway and so on). Unfortunately, this also varies from country to country along with standards. For example, the PTT (Post, Telephone and Telegraphic) Systems installed so far in the U.K. (PRESTEL), West Germany and Austria (BILDSCHIRMTEXT), Holland (VIDITEL), France (TELETEL), Japan (CAPTAIN), the United States (VIEWTRON and COMPUSERVE), Canada (TELIDON) etc. all have their own variations and standards are obviously of paramount importance with such a plethora of systems and equipment. To this end, the more enlightened suppliers have adopted the CEPT (Conference of European Posts and Telecommunications) Standards. At present, this might appear to be the only logical standard to be implemented anyway, owing to the difficulty faced on a Worldwide basis by the CCITT (Consultative Committee for International Telegraphy and Telephony) as other standards attempt to become adopted internationally, for example, NAPLPS (North American Presentation Level Protocol Syntax) with their own champions such as Telidon or Captain.

But back to the U.K., where most of Air Call's experience lies and which probably has (along with the Netherlands) the most comprehensively implemented PTT Systems in the World. Here, Videotex seems set to take off in a big way.

Cont./...

Viewdata is a screen-based information handling system, which at worst is used for text retrieval and display and at best can replace conventional transaction processing monitors. It employs a separate database of pages and frames (or databases in the case of Viewdata/3000) which may or may not be interactive with the normal computer database files, dependent on the sophistication of the implementation. These pages can then be accessed in a variety of ways by Viewdata (i.e. Prestel-Standard) terminals which, because of their similarity to normal colour television receivers, are far more acceptable and less expensive than the usual computer terminal.

Although it is estimated only about 150 private Viewdata Systems have been installed to date, the low number being at least partially attributable to the reasons I outlined previously, there are probably over 20,000 users of Prestel and a quarter of a million teletext sets, not to mention the thousands of bureau users of Viewdata Systems. Up to now, confusion seems to have existed over the relative merits of a public or bureau-based viewdata system against private viewdata systems, but with the advent of large multi-national companies showing a real interest in Viewdata Systems, private Viewdata Systems seem set to challenge traditional processing techniques with the present state-of-the-art technology they now employ.

A good representative of this state-of-the-art technology is Viewdata/3000, a mini computer based Private Viewdata System. Designed by Nova Computer Software in Holland, it is one of the most powerful systems available today, a fact which is due to the unique combination of the Hewlett-Packard 3000 processors (running MPE) on which it runs and the design of the software. It has been implemented in Great Britain, Germany, Switzerland and Holland and, as such, is being constantly updated and revised (in an upwards-compatible form) to ensure that it is in the forefront of Viewdata Technology.

Cont./...

Air Call Computer Systems decided to adopt it as a basis for its future product range of applications systems, because as a totally user orientated information system, Viewdata/3000 is the embodiment of Air Call's commitment to providing complete customer service through 'state of the art' technology. Visually, the impact of Viewdata is attractive which, with its simplistic and foolproof operation, overcomes any reluctance there may be to extend data processing into all or any parts of your organisation, even employees homes if required.

This is because Viewdata makes use of a normal colour television receiver, rather than an expensive and highly specialised Computer terminal. At present, the cost of such terminals and the necessary complexity of communications continues to be a major constraint to the large scale dispersal of data processing facilities, but with Viewdata and taking advantage of its latest programming technology, you can access, at the touch of a button, information in ways which hitherto only skilled data processing professionals were able to do. In addition, the Viewdata Sets will give you the facility to access not only your own company information over the private Viewdata system, but also access public (PTT) systems such as those mentioned previously. Never before have you had the capability to obtain so much valuable information in such an easy fashion, presented to you in a variety of attractive colours and graphics.

But obtaining information is only part of the answer to providing a really effective business system. The supply and processing of raw data is equally as vital ("garbage in means garbage out"). This is where VIEWDATA/3000 comes into its own, in that the information required can be collected in exactly the same simplistic fashion, making the task for you enjoyable and attractive (Certainly better than form filling, entering columns of figures in a ledger or other tedious activities).

Cont./...

In addition, apart from providing set information (an "Information Provider" or I.P. activity) this data will automatically be processed by the central computer where necessary, to provide you with what you need when you need it. The whole process is so advanced that it required no user intervention.

This whole area, that is the integration of Viewdata Systems with normal data processing systems, is perhaps where the greatest progress has been made in transforming Viewdata into a really effective vehicle for all business information activities, and a serious contender for the future basis of data processing. Quite simply, Viewdata 3000 (for example) is fully interactive in the way DP people understand and need, i.e. on a field by field basis rather than (as is the tradition) page by page. It is no longer necessary to regard it as having to be completely separated from the main computer network, which is the reason for my earlier comment, that Air Call decided to adopt is as a basis for its range of applications systems.

Furthermore, coding for the design of screens can be embedded in the applications programs which can be written in any language, so that screens can be created under the control of the applications programs. This would allow, for example, bar charts to be created directly as a result of an analysis report by the applications system on the main computer.

The requirement for sophisticated and powerful information handling techniques by the computer, led Air Call to select the Hewlett-Packard 3000 as the central machine for its VIEWDATA/3000 Service. The primary criteria for this selection included the advanced database management techniques (e.g. IMAGE) available on this machine, an essential for the constantly changing pool of information upon which VIEWDATA/3000 relies, and the reliability and ease of operation of the system (many VIEWDATA/3000 users wish the service to be available 24 hours a day with little or no specialist staffing).

Cont./...

One of the major constraints which has become apparent as more and more companies begin to use Viewdata on a large scale, is the number of users which can be run concurrently with an acceptable response time. Because there is no software limit on the number of users, the HP3000 was, again, a natural choice as (under MPE V) it will be able to run over 1000 users concurrently and allow over 3000 sets to be connected to the network. We have found this to be a major advantage in providing cost-effective Viewdata networks, especially as the software itself is very efficient in CPU usage and memory resources.

The Viewdata terminals need only a telephone line and mains connection to give you access to all these facilities. Consequently, they can be used with equal effect at home, in the office or out in the field. This can allow you to implement a vast number of applications which were not cost justifiable or even technologically possible before.

Our experience with modern Viewdata terminals (most of our users prefer the Philips 3607 or Sony KTX9000) is that the facilities considered a luxury only a couple of years ago, are now a prerequisite. This includes; off-line editing and page preparation in order to reduce line costs; local storage; automatic dialling and identification as many users want the same sets to access different Viewdata Systems easily but securely; automatic acquisition of pages; and selective transmission of data from pages in order to enhance response times.

Because Viewdata/3000 is one of the few systems to be designed in accordance with the CEPT Standards, it naturally fully complies with the British PTT requirements. This system is probably the most widely used in the World and amongst the most sophisticated, being able to offer advanced facilities such as access from one private viewdata system or set to another computer via GATEWAY. This function is also fully available in VIEWDATA/3000 which, along with the most comprehensive list of features of any Videotex System, make it a natural choice for any organisation with a Dynamic Information Environment, the requirement for a constant flow of up-to-the-minute information amongst its staff.

Cont./...

The concept of a Dynamic Information Environment is, of course, not new to businesses. It has long been the objective of many data processing departments to achieve a freely operated, up-to-the-minute interchange of their Company's information amongst all their users. With the latest enhancements to Viewdata/3000, we see no reason why this cannot now become a practical reality. A natural feature of such a system would include sophisticated electronic mail and message switching, as well as comprehensive security, closed user groups, user statistics and sufficiently advanced editing to allow users the means to get their information and designs onto the screen as quickly and easily as possible.

For the reasons outlined so far, the traditional users of Viewdata Systems have been large· organisations with a need for constant access to fairly static information. However, with the advent of these enhancements and particularly that of true interactivity with applications systems, the scope is widening. Pharmaceutical firms, motor dealers and distributors, insurance companies, retailers, travel agents, banking and finance institutions are all beginning to recognise the benefits of Viewdata within an established data processing background, while other industries have started investigations into their future DP policy involving Viewdata as its basis. Bureau applications of Viewdata also have a great appeal, especially for first time users, because while the costs are comparable to owning a microcomputer, no programming or operational expertise is required. As an example of the versatility of this new medium, microcomputers too are in many cases able to access Viewdata systems via Micronet and we can increasingly see the use of Viewdata as a communications protocol.

VIEWDATA/3000 is a major area of expertise within Air Call and as such we are one of the few computer systems companies to offer a complete service to viewdata users. These services range from bureau applications with perhaps two or three terminals to the supply, implementation and on going support of complete networks, including the computer, telephone links, modems, viewdata terminal controllers, viewdata sets, and the viewdata applications systems.

Cont./...

The support would cover all aspects of your system from the design and supply of the applications software to training, maintenance and updating/modification of your systems as your company grows.

For this reason, Air Call will shortly be releasing a complete range of Viewdata-based applications systems aimed at meeting the requirements of not only specific market places which will benefit from the unique concept of Viewdata, but also at general purpose systems to benefit the first time user.

For further information on Viewdata/3000 and Viewdata based applications Systems, please contact the author on 0582 603123 or on Prestel account number 582609925.

DCB/CS/01.08.83

systemsolve

ELECTRONIC NEWSROOM SYSTEM

OVERVIEW

ENS, a highly resilient electronic newsroom system, was created by the BBC to support its prestigious early morning programme BREAKFAST TIME. Designed by journalists for journalists, the system was developed by SYSTEMSOLVE based on HEWLETT PACKARD hardware. Although ENS was originally targeted for the broadcasting environment it will support any organisation that demands scripting and production scheduling.

In the newsroom, ENS sustains a 24 hr production cycle be it for news or features and handles every production need from programme planning to generating Autocue on transmission. Its design speeds production, extends deadlines, facilitates on-air decision making and permits instant script changes.

It processes a 100 sequence programme, as in BREAKFAST TIME, as easily as a 20 sequençe bulletin and holds 500 pages of script if required. Its automation allows everyone working on the system instant access to any part of it.

Based on a modular principle it can stand alone on a single site or be extended to link a network of stations that share common sources or a common output. Similarly this approach will allow tailormade facilities to be easily added, as required.

Typically it can reduce staffing levels by 10% and an installation is projected to pay for itself within five years.

EXECUTIVE PRODUCER

ENS allows the Executive Producer finger tip control of the programme. Without moving from his desk he can access every part of it from his VDU keyboard. He can use the electronics to speed his production process and extend his decision deadlines. He can be assured that the software prevents unauthorised interference with his decisions and the inbuilt hardware resilience will safeguard his programme.

In planning the programme he can access electronic Diary, Agency Copy and "Shelf" information files. To assist him in its productions, a Storyboard lists the items to be included in it together with the producers assigned to work on them. Further into the schedule, a Sequence facility allows him to juggle all his programme items until he achieves the right mix. At the same time additional features allow him to assign presenters, approve scrips and, if required, automatically produce the formats of standard recurring items.

PROGRAMME PLANNING

The Dairy can hold an unlimited number of entries for up to 365 days from the current date. Each entry may be easily cross referenced and indexed under any four of sixty category headings and be instantly accessed from a master menu. Entries may be transferred to a different category or date as required. With the ENS diary bulky desk diaries and drawers of card index files are a thing of the past.

The Prospects facility allows the Planning Producer on any given day to sub-edit all Diary entries to those most likely to be transmitted.

ENS automatically stores up to 12 international or national News Agency wires as well as holding space for copy from programme correspondents to be input from the production office. Each story from each Agency is filed and listed in reverse time order but slug searches can be linked and carried out both forwards and backwards in time.

The "Shelf" is a store for any script written for transmission but not immediately required. A Shelf search can display not only the script text but also camera directions, briefing material and any important information necessary to get the item on the air.

The Storyboard is the end of the planning process and replaces the production office wall "marker board". On it are displayed all the items selected for possible inclusion in the programme together with their assigned Producers, identifiers (see below) and estimated durations. It can be universally accessed but can only be input and amended by the Executive Producer.

PROGRAMME PRODUCTION

The "Skeleton" facility commands the computer to display day to day all the formats of standard programme items. The principle advantage of this facility is to avoid programme secretaries needlessly typing and retyping the same information: ENS can do it quickly and more efficiently.

The Sequence facility displays the "standard" Skeleton items together with the "new" Storyboard items and allows the Executive Producer to juggle with them until he achieves his desired programme structure. The facility may be viewed universally but only amended by the Executive Producer.

Script Approve allows the Executive Producer to view and approve scripts either en masse or individually. Software checks ensure that no script may be printed for transmission or transferred to Autocue without being approved.

Presenter Assign again allows the Executive Producer the ability to allot items to particular programme presenters either en masse or individually. At any time assignments may be altered to meet new programme requirements.

The automatic calculation/recalculation facility built in to ENS allows the Executive Producer the ability to know instantly his running time and highlights segments of his programme that are either overrunning or underrunning.

PRODUCER

The Producer loses none of his creative independence with ENS. Software control ensures that no-one may eavesdrop or overlook his work until it is completed. Furthermore the silent keyboards of ENS significantly replace the clattering of typewriter keys and contribute to the relaxed environment that the system seems to engender.

Producers too, like the way that ENS allows them fingertip access to sources of information thereby saving them from continual journeys to teleprinter rooms or libraries. Writers input their own material and no longer have to reply on programme secretaries following their drafts.

With ENS, Producers can work as easily on location as back at base. All that is required for them to do their job is a terminal linked to a standard telephone line.

The ENS text editor provides a standard range of script writing facilities but features special modifications for broadcasters. Default margins prevent producers from overtyping in columns where Autocue cannot read and unique copying facilities allow script to be duplicated where required rather than laboriously retyped.

View Script allows any member of the production team to scan script text once it has been released.

Scrip Edit permits authorised personnel the ability to edit text as required.

A "Profile" is created for each script. It files essential information about it and controls its progress. It provides space for Producers to input camera directions, comprehensive briefings for Presenters and features a code, the "Identifier", that follows the script throughout its production stages.

It can be viewed and accessed by every member of the production team. Profiles replace paper "kits" and person-to-person briefings that may be difficult, or even impossible, on a day-long production cycle.

DIRECTOR

ENS allows the Director to be more creative and releases him from a lot of the tedious administrative work that goes with his job. Directors now enhance the shotlists that producers have input on their profile screens and leave the computer to automatically sequence the technical detail in accordance with the main sequence order.

With the VT log facility the Director controls one of the more important administrative facilities in the system.

The Technical Sequence automatically scans the first pages of all script profiles and arranges them according to the main sequence order. The information input by Producers and updated by the Programme Director will then convey to the technicians the actual order of camera, vt and graphics sources. Given that the Technical Sequence itself could be extensive, a search facility will pinpoint the desired detail.

The VT LOG facility allows the Director and his assistants to keep track of the VT available to the programme and the items appearing on any particular tape.

PRODUCTION NEWSROOM SYSTEM

ENS liberates the Production Assistant and Secretary. No longer are they just the copy typists of the newsroom but, through control of the Autocue and Printing facilities, now assume a more integral role in the production. With the technological innovations of ENS they find that the messier side of script production, is a thing of the past: no script retyping for Producers, no carbonised paper, no stencils and no frantic last minute script collation.

The ENS prompt facility, AUTOCUE, automatically locks on to the scripts in the computer and transfers them electronically to the camera head mounts. It is a major feature of ENS not only providing front line resilience for the system but is also able to support it in a stand alone mode. It features reverse phase text, variable typefonts and remote operation.

The Printing facility allows secretaries and, for that matter, anyone on the production staff the ability to print Technical Sequence, Scripts Dairy and Skeleton pages. Paper is an important part in the ENS concept since it is assumed that no production in the foreseeable future will be able to function properly without it.

PRESENTERS

ENS allows Presenters the ability to brief themselves automatically on arrival in the production office - very possibly much closer to transmission than hitherto.

The system will print for them every script, profile and message pertinent to their individual assignments in the programme and allow them to change text with a minimum of fuss. They will be happy to know that any script mutually agreed with a Producer will appear exactly as written on the studio prompt device - and never need to check Autocue before transmission.

The Briefing facility is the automatic printing of scripts and profiles and is activated by the Presenters own passwords. It may be essential if Producers or Researchers have been re-assigned to other stories and are just not available in the newsroom.

MAIL/MESSAGE

ENS gives every member of the team his or her own comprehensive office facility. In it they can receive and send mail, file memos, create private administration areas and change their passwords at will.

Additional features give them the ability to designate to colleagues the power to open and reply to mail on their behalf when either sick or on vacation.

MAKING SENSE OF OFFICE AUTOMATION

TREVOR WING

OFFICE PRODUCTIVITY DIVISION

I'd like to spend the next 40 minutes talking about what Hewlett-Packard has learned about office systems over the past 3 years.

I'd like to start off by talking about the seven key ingredients we feel are important when considering implementation of an office system, or an office automation network.

Office Personnel

These are the people who are going to use the system and it is obviously important to think about the needs of those people because without their commitment and motivation, the installation is not going to be successful.

Workstations

These are the tools that those people will use to interface with the system services.

Office Services

These are the services behind those workstations, such as Word Processing, Electronic Mail, Business Graphics, Data Analysis, etc.

Work Objects

The work objects are the items that those facilities operate on. So, in the case of Word Processing, the work object would be a document; in the case of Business Graphics Application, the work object would be a graph; and in the case of Data Analysis, the work object would be data extracted from a database.

Networking

Networking is an ingredient which is becoming more and more important. If you start with a small office system, networking probably is not going to be high on your list of priorities, but as the system grows into a larger implemenetation, covering multiple departments and buildings, the networking is going to be a major consideration.

Implementation Assistance and User Training These are the services which we add to the products to help you understand where you need these facilities, how to implement them succesfully, how to train the users, and finally, on going support for the system.

What we will do for the next 30 minutes or so, is take a look at each one of these areas in a bit more detail. Hopefully, I will raise a few issues that you probably should think about if you are thinking about implementing an office system.

Office Personnel

We think you can generally classify personnel into three different types of people

Management People
Secretarial People
Professional People

Although a gross generalization, as far as office facilities are concerned, these seem to be the terms that fit. They are the people who are going to use these services.

Workstations

The access point can be generalized into two areas:

Office based -	those which you would use inside your company from your own desks, the fixed type workstations.
And	
Portable -	portable workstations for people that spend time travelling. People that need access from home or people who want to use a personal computer as a portable access point for office systems.

We see each of these as being as important as the other. Really, it is a case of where you start from, or what type of person you want to give an office service to. In some areas, the type of things you would need from a management workstation would probably. be different from a secretarial workstation. In some areas they would probably be the same.

Travelling Access

This is really a lightweight portable access that you can carry around with you. This also consists of personal computers for those who need individual computing applications on top of their office services.

Office Services

We do not really feel that there is one particular office workstation that we would like to promote as an office station; we think there is a variety of them. Each of these should be able to access what we call the Core Functions. The Core Functions in our terms are:

Electronic Mail. This is to communicate items, whether it be a document or a business graph.

Personal Filing. Personal filing is to organize those items you are dealing with, whether it be a document, graph, piece of data, or whatever.

Time Management. This is the diary/calendar facilities. Most people in your organization are going to need a type of Time Management. Everyone has a diary - this is an automated version.

Simple word processing is distinctly different from full-function word processing. Full-function word processing is the type of word processing that a secretary needs if she is going to spend much of the working day typing. The simple word processing in the Core Functions is the type of word processing that one can learn very quickly and would cater for probably 80% of most Manager/Professionals needs.

These Core Functions need to be accessible from each of these different types of workstations.

Further support tools are provided which are specific to the type of worker.

Full-Function Word Processing. Full-function word processing is really for secretaries spending much of their time typing.

Business Graphics. Business grahics is for people that need to work with graphs on a day-to-day basis, using access data and plotting graphs etc. - decision support tools.

Community Filing. Those facilities we think would be managed by departmental responsible persons - secretarial-clerks. They would be responsible for managing a group of files rather like the traditional filing cabinet that you keep in your office.

Data Access. This service allows you to easily access and manipulate data stored in data bases in your systems. We think that is quite important, as a lot of decision making is based on data that is in your organization right now. A lot of it is computer based.

Integrated Printing. There is not much point in being able to deal with graphs, word and data unless you can integrate them and print them. We think an integrated facility is also necessary to be able to merge these items together and put them onto one piece of paper.

Two areas here that have not really become a part of office systems, although probably will in the next four or five years, are Voice Capture and Image Capture.

Voice Capture is actually capturing the spoken word and dealing with it as an object.

Image Capture is the picking up of things, like a newspaper article, that you do not want to re-key, but have a use for that piece of information.

Work Objects

Those office services, as said earlier, access work object. If you boil down the work objects from those facilities, you find that they are words, data and graphics. Each one of those is as important, we think, as each other one (and in the future voice and images). So, the office system you choose needs to be able to deal with these object types and have the facility to grow into dealing with future office objects (voice and images). Right now, most workstations are connected by a piece of wire which is an RS232 style connection. I think what we are going to see there, as time goes on, is a connection of workstations via PABX's, and that sort of thing, so you can use your telephone point to connect your workstation to your office system. So, workstation system is something that everyone will need.

Networking

System-to-system Single Vendor, is really letting the facility grow into a larger organization. Hewlett-Packard is a good example; we now have 3000 users of our office products world-wide, and if we could not distribute those over more than one system, we would find that we could not usefully do the job. So, enlarging the community really needs system-to-system networking. As time goes on, we will need to address multi-vendor networking so that we can, in fact, connect our systems to other peoples systems.

So, those are really our views on those seven points.

I would like to bring out those points a bit further and raise some issues that you should consider in each of those areas.

Office Personnel

Office personnel are really everywhere in your business, and if you think about yourselves as users - the people you work with - they are all really office personnel. No matter where you are in a company, you will find that there are office workers; secretaries, principles, professionals and managers. We do not think that office systems are confined to any particular area of a company. Wherever you may find someone working in an office, we think that they have a need for an office system. So, we see office systems spread around companies quite liberally.

The major cost, in business, in terms of labour costs is in professional and managerial people. In the past four or five years, most office systems have really been slanted towards the secretaries and typists. We think that if you are going to get some real productivity benefits, in terms of saving those people's time, you also want to effect these people as well as the other types of people. We feel that our office systems, and really any office systems, should be able to address the needs of not only secretaries in the organization, but also professionals, managers and clerks. Of course, if you think about it, that means that we need different sorts of services.

Some of the issues we feel you need to consider if you are implementing an office system in your company today, as far as office personnel is concerned, are really:

> Who do you start with?
> Do you start with the secretaries, do you give them full-function word processors first, or would it make more sense to put an electronic mail system in to implement, to improve the productivity of your professionals, managers?
> Where do you really start?

So, those are issues that you really need to consider. It will be different for many companies; there is not any set answer those questions, but they are issues that need to be considered.

Should you do a pilot first? Should you implement a small system for a limited number of people and evaluate the benefits of that pilot before you go into a full production system? Many of our customers do and some do not; it is a personal issue.

How do you train the personnel? What sort of methods does the vendor offer you in terms of training the people who want to use these systems? Is it a long lease-cycle? Is it easy? What is really involved in training the people. Also, what internal support is required? What do you do as customer, in terms of supporting these systems that you are installing? There are many more issues, but these are just four I have brought up that are pretty important.

Office Workstations

I have listed the facilities I think that each of the workstations above should provide. Below are some Key Core Facilities.

Electronic Mail is a key core facility because it moves items between people, time management, personal filing and simple word procesing we think all of those types of people would have a use for. In addition, depending on the types of people, we would see some things there that are unique to a type of job (e.g. in the management area, ad-hoc data reporting, - the ability to get at data to help with the ability of decision making to improve the process of decision making, - is not relevent to a secretary, but is very relevant to a manager.

Full-function word processing would be for a secretary because she will probably be spending a lot of time typing and will need the facility to carry out those tasks. A manager would not, of course, need a full-function word processor. So that is one area where the "bolt ons", that is, on top of the basic office services are unique to the people within the organization.

In the professional area, the real difference is job-related computing. Professionals in most walks of a company will find that they have some sort of use for a job-related computing task. We think the extra ingredient for professionals, over and above the basics, would be job-related computing and business graphics. It is important that the core functions are accessed from all those workstations because you will not want to worry whether the person's workstation can access electronic mail or time management. Any workstation you choose should be able to access those facilities. Of course, if you need a special workstation for full function word processing, using the facilities in an advanced terminal to improve that facility, we still think it should be able to do the basics. It should be a super/sub-set strategy; sub-set for all the core functions and super-set where you need the objects for a specific function.

In the portable area, we are thinking about what you would need there, travelling terminals for home or mobile access. If you are out of the office a lot - you spend a lot of the time in the field (a salesman would be a good example of that) - then you need some sort of portable access. Again, we think it should be able to handle the core functions. These sort of terminals are not usually very sophisticated as it would be difficult to find a terminal that is portable and also a full function word terminal. So, the core functions are the things that need to be accessed from a portable terminal.

Personal Computers - The real difference is that they handle individual computing needs, CPM based systems, Unix based systems, wherever you have a need for local computing which is not handled on your office services machine. Of course, on top of that, we still feel that you should be able to access the core functions, you still want to exchange items with other people, you still want to use it as a mail terminal, you still want to have your diary running there. So, the set of core functions still crop up even on a personal computer. It would not be very useful to have a personal computer that could not access those core functions.

So, the issues on workstations that you should consider are:

> Who needs a workstation, first of all?
> The view right now in industry, is that everyone should have a workstation on their desk. This may be true in times to come, but right now, you need to examine very closely who actually needs a workstation in their job. It is going to be a gradual, growing process.
>
> Which type of workstation for which type of person?

Once you have decided that you are going to give a person a workstation how would you decide which one is best for them? Should you share a terminal amongst a group of people, or do they need to be accessed to it so often that it should be dedicated to only one per terminal? This will differ depending on what sort of company you are and your committment to offices systems in terms of cost.

Who needs portable access?
Does anyone in your organization need to have home access? Do you have salesmen who could really benefit from a portable access and could make good use of it?

Office Services

Just to reinforce the points made throughout, we think there is a core group of services which spans most job types and have given six examples of different types of business in a company.

Engineering
Marketing
Business Alloys
Analysis
Manufacturing
R & D
Data Processing

We think these core functions span all of those areas. All the people in those businesses or areas of your business can gain benefits by using these core functions. On top of this, we see some job related functions which should bolt on to that. For instance, you probably would find that the people in the business analysis area would find business graphics very useful. We think that it would be easy to bolt these on top of those functions. If you look at bringing the office services more down to earth, what we are really talking about is a set of desk management facilities - a term we have coined at Hewlett-Packard to describe these core functions:

Electronic Mail
Basic Word Processing
Time/Calendar Management
Personal Filing

We think that these are the real core of an office system on top of that, we have the job related bolt on facilities, which, of course, not everyone will need but can choose if necessary:

Word Processing - Traditional area for full function secretarial style access.

Management Reporting - To help decision making.

Business Graphics - To put that data in a more meaningful format.

Laser Printing - Lasers are really the only technology we have to offer in industry today that can truly integrate all of those work objects together and print them.

Personal Computing - People that choose personal computers really want to have access to this core, even if it is only once a week, they still need to do so.

Personal List Management - Ability to manage a set of records personally, e.g. business card index, telephone numbers.

These are the ingredients we think you should consider when looking at Office Services.

Workstations

The issues given on Office Services are similar to Office Workstations - who needs what? Does everybody need everything? What should you give people in terms of Office Services or facilities? How do they integrate - how easy is it to take a graph you have created with business graphcs to file it, mail it to someone else and then reproduce it at the other end? Therefore, is it smooth to integrate or is it jerky, which would mean that people would not use it? Therefore, integration of the facilities from the users point of view is of prime importance. Can the work items be passed between services; if you choose a work station that is going to support business graphics can it support the core services that go alongside it?

Work Objects

The issues on Work Objects are:

Firstly, of paramount importance, can the system handle all three - words, data and graphics? Can they be integrated or put together and be sent to somebody else so they can then print it? A good example here is a report, as it is important in a report to be able to add grphics and have it printed as one. Can all three be mailed; can the mail system, which moves items between people move all three types of items or is it a messaging style system? Our belief is that it should be able to move all three plus the extra two that industry is going to add in the future.

Can all three be filed? Can you manage all three objects; can you organize them in your filing cabinet and can you access them by subject name regardless of whether they are words, data or graphics? Can all three be merged and put together as one on an output as already stated?

These are the questions that would give a good feel for a good quality office system.

Networking

Everybody does have a need for networking. If you look at a simple Principle/Secretarial relationship in a traditional office, it was really a case of the intray and outray. It was possible to put things in your outray and have them delivered to your secretary or one of your peers, and things delivered in your intray were delivered from other people. We do not think office systems are any different. In a work group on the same system, you should be able to communicate items between each other on or between different systems in the same facility. The key words here are user transparent. It does not matter whether you are sending to three or twenty people within the corporation, it should be as user transparent as the three people in one small work group. Also, one must be able to communicate with other systems, other hosts (i.e. IBM), because we find that a lot of people we sell to need to communicate with an IBM compatible host, which we think is quite important.

Issues on Networking: One of paramount importance is: Is the networking transparent to the user? If I should send an item to someone else in my company, is it handled for me, am I shielded from the system that does it for me? This is a key issue. Am I also shielded from the system if I send it to another system; is the networking transparent regardless of the network size?

Links that are supported between systems and what is supported are quite important. Is there direct connect between two systems, dialup between two systems and can it autodial so it takes you away from having to think about dialing? Can you use the PSSX.25 (packet switch) network? Is it possible to use private wire, and is there a telex option to use between systems? Host access - is it possible to get to the Standard IBM formats 3270, to be able to emulate a terminal running on an IBM? This is quite important to managers if they have a corporate data base, as they want to be able to use their same terminal that you are selling them as an office systems workstation to access their central main frame, which is actually 3270 emulation (also 2780 and 3780 if you want to communicate with systems in that way).

Implementation Assistance

What do you get from Implementation Assistance? We think that there are two important things, the first one being the feasibility study, which is a service carried out by your selected vendor, or many vendors, before you install your system or even commit to your system. The sort of things you need to address here are; where can office systems be used in your company? It may not be apparent where you can actually make use of them in some areas. What are the benefits? Can you qualify the benefits of installing those systems? Can you justify the investment? If you cannot justify the investment, you are probably better off not doing it. Exactly who needs what - who do you want to give what sort of service to, and where? These are the things we feel a feasibility study should address.

Moving onto the implementation plan - Once you have decided which vendor you are going to use - how do you implement it? Do you do a pilot first? What sort of hardware is needed? What sort of software is needed? What sort of user training are you going to commit to? How are you going to train those users? Are you going to make a phased implementation, or are you going to install from day one with the community you have decided? How would you qualify and measure the benefits after you have installed it? Those are the things we feel you should get from an implementation plan.

Training and Support

Two key issues after you have chosen the system are training and support. How do you train the people? Is it a classroom course whereby you would have to send your people away to a classroom? Does it have computer based facilities, can those people learn the product from their own terminal on their desk? Self-paced - are you going to give them a book, or is the vendor going to give them a book to work through rather like a classroom exercise book? Are you going to send your people to a training center or can the training center effectively be brought to you for on-site training? Is it possible to train an instructor from your own company to carry out the training?

Support

This is making the thing work after it has been installed. This comes down to hardware, making the thing run - how reliable will it be to run, how easy and quickly to fix should something go wrong, updates of software, upgrades to the products as time goes on, upgrades to the manuals and do you have somewhere you can get information?

Question: Can you enlarge on the term "Electronic Mail"?

Answer: Our meaning of Electronic Mail is two fold; first, it is an effective aid to management by getting data to them quicker in a more useful manner; second, to use electronic mail as an office service to move items between people. By this we mean any item, if it is a secretarial word processing document or business graph, the mail system should be able to move that to someone else. Mail is really a transport service but it can also improve management effectiveness by allowing a group to communicate more freely and more effectively between themselves.

Question: How do you define the difference between secretarial word processor and a management word processor? How do you decide which one makes best sense?

Answer: Our view is that anyone should be able to use a basic word processor with probably 15 minutes of instruction. The point being that if you are using a sophisticated word processor once a week, you are probably going to forget what you have been taught. With a simple word processor that you could pick up in 15 minutes, it would be very usable if you go back to it in a week's time. We feel that this is probably enough for people who are not going to spend a lot of time typing, or even a secretary who does not spend a lot of time typing. However, when you get into the realms of doing a lot of typing (i.e. large documents), you really need more facilities and therefore more sophistication and more training. So what we would say is that none of these have a specific target user as it depends on the job itself. If they do not spend a lot of time typing, then a simple basic word processor is probably what they would be using. If a lot of typing is involved, then we would advise a more sophisticated full-function word processor.

Question: What exactly is a manager going to use a word processor for, is ge going to add final points to a memo, type reports, or what?

Answer: I would say that depending on the manager, probably any one of those. At the basic end, he would probably format a mail message if he wants it to look nice. At the other end, he could quite easily produce a report with a basic word processor, although it would not be that useful if you wanted to spend a lot of time chopping it around. The main issue is to make it as simple as possible to use, and then the user can use it as he sees fit.

Question: If you have a good set of support software services for your systems, why do you need training?

Answer: The core function area does not need very much training and can be computer based, certainly with our products and within an hour of using the computer based learning you can mail a message, edit it itc. When it comes to full-function word processing or business graphics, there is quite a lot more to be learned to make full use of these products. that is hwere we would recommend user training.

Hewlett-Packard Limited

Office Systems Pinewood

The Architecture of Integrated Office Systems.

Peter Williams

Nine Mile Ride, Easthampstead, Wokingham,
Berkshire, England. RG11 3LL
Phone Crowthorne (0344) 773199

CONTENTS

1. Abstract..2
2. Introduction..3
2.1. Availability and Convenience Integration......................4
2.2. Data Communication and Exchange...............................4
2.3. Data Integration..5
2.4. Functional Integration..5
2.5. Integration of User Tasks.....................................6
3. Current Architectures...8
3.1. POST Integration..8
3.2. PRE Integration...9
4. The User Model...10
4.1. What IS a User Model ?.......................................10
4.2. Why the Model is Important...................................10
4.3. Existing User Models...11
4.4. Integrated System Models.....................................12
5. Architecture...13
5.1. Object Based Architectures...................................13
5.2. Components of the Architecture...............................13
5.3. Implementing the User Model..................................14
5.4. User Scenario..15
6. Conclusion...19

1. Abstract.

This paper will look at a particular challenge facing the designers of automated office systems - INTEGRATION. Office systems need to be integrated both amongst themselves, and with other applications. The paper will examine what it means to be integrated, why we need to do it, and what the user is going to think about it. It suggests an architectural approach which seems to fit the requirements.

2. Introduction.

The dictionary defines "TO INTEGRATE" in several ways which may or may not be relevant to our discussion. It means "to combine into a whole"; "to complete by the addition of parts"; or "to bring into equal membership".

There's something of all these definitions in what we intuitively understand by "integration" of office systems. The concept is rather richer when applied to office systems than in the more abstract world of conventional data processing applications, as we shall shortly see. The same ideas are present, though, and we may be able to reuse some of the techniques we used in the past.

If I say that I have integrated order entry with sales analysis, most people who understood what I meant at all would not assume that I had written a program that did both of these. They would naturally, and in most cases correctly, assume that I had managed to arrange the data so that both applications could use a common data bank, and that my order entry application had possibly been augmented to assist the sales analysis function.

But if I say that I have integrated time management functions into my electronic mail system, most people would assume that both functions were available from a "single" application program. What am I assumed to mean when I say that I have also integrated word processing into electronic mail ? Presumably, I will not only have enabled you to create documents with my word processor, but that the electronic mail system is also able to distribute them to other users.

So right at the outset, it's clear that we mean different things by the same term, and we allow the integrator (me) some latitude of interpretation. This may be playing into my hands !

So let's qualify the term a little, and then look at what people need and expect of an integrated system. That way, we may be able to recognise what we have done before, what we are doing now, and how we can fulfil our customer's expectations in the future.

Looking at the development of application systems and especially office applications, I think we can detect 4 types of activity which you can legitimately call "integration". I shall call them:

1. Availability and convenience.

2. Data communication and exchange.

3. Data integration.

4. Functional integration.

2.1. Availability and Convenience Integration.

It's sometimes irritating to use a product that takes a long time to start up, but will not let you have access to any other system facilities without shutting it down. You will find, therefore, that a lot of products will let you issue some MPE commands while you are running them. Sometimes, the products would be very hard to use if they didn't offer this facility. For example, if you need the name of an MPE file to edit, it's tedious if your word processor won't show you a list of files.

So you could say that such products are "integrated" with MPE in the sense that you have access to the facilities of MPE while using them. The integration is for your convenience. Neither MPE nor the word processor is particularly dependent on each other - not for the purpose of supporting THIS level of integration, in any case.

Convenience integration is quite easy to do, and is therefore very cost effective for the developer. MPE will let you run almost any program from any other, so it's possible to do a good job with this sort of integration on the 3000. Most HP applications will do this much for you.

We even have a product built around this ability in HPMenu. This is a program which runs other programs for you, and enables you to configure your own application environment. HPMenu therefore integrates every application program it's possible to run on the 3000 - or it could do, if you asked it.

2.2. Data Communication and Exchange.

Convenience integration is sometimes not enough. It doesn't, by itself, tackle the expectation that a word processor integrated with an electronic mail system ought to be able to distribute documents very easily.

The integration of sundry applications with HPDeskManager is an example of data communication and exchange between separate applications. The two applications communicate data between themselves, and, to some extent, KNOW they are doing so.

So, as you'd expect, this level of integration cannot be achieved without changing the applications. They need some protocol for communicating. In HPDesk, this protocol isn't symmetric. So, whereas HPDesk can cause a document to be created, and then mail it, HPSlate cannot today cause a document it has created itself to be mailed. And it's not very likely that it ever will.

Trying to integrate all office applications at a "peer to peer" level is an interesting challenge, but doesn't produce an "integrated" system for the end user. From the user's point of view, there shouldn't appear to BE two products, just a lot of data and a lot of functions. It's not a very cost effective approach for the developer to extend the user interface of every "product" to incorporate all the reasonable functions of every other. He's only going to end up with a lot of niggling inconsistencies, and users who aren't as grateful as he'd like !

But HPDesk can do a good job of integrating a large chunk of today's office functionality using simple data exchange protocols - rather a lot of them, but none overwhelmingly difficult. This is because HPDesk is designed to take advantage of anything other applications can produce for it. It's a parasitic program, in a way. As we shall shortly see, though, it has something else going for it - it's an integrator of data.

2.3. Data Integration.

When two or more applications use the same stored data structures, they are integrated at the data level. Note that this isn't quite the same as saying that two products can communicate data between them, for in this latter case, a communication protocol is required.

Conventional data processing applications furnish richer examples of data integration than does our automated office today. Anywhere an organisation sets up a full blown Data Base Management System, it is presumably doing so to achieve data integration of a family of applications.

In the office today, we can integrate text and graphical information in documents. That means that several applications - the word processor, our graphics editors, and our printer software - create and share a common data format. They are integrated at the data level.

We might have achieved this effect in a variety of ways. It's interesting to look at how we actually do it. We didn't alter the graphics products at all. We changed the word processor to understand what the pictorial data structure looks like and to plant a reference to it in the text of the document. And we needed a laser printer procedure which would understand how to print the text and the picture on the same page. Only the laser printer software is really feeling the effect of this data integration.

We might have decided to amend the word processor to enable pictures to be designed and drawn with the text, using graphics software. We might have decided to extend the graphics editor to use the word processor to handle the text bits, using graphical operations to position them on the page.

We have opted for now for a simple approach which enables the text editors to plant references to graphical data in their own data, and wait for the hard copy before realising the data integration. We have, in other words, not opted yet for functional integration of our text and graphic editors.

2.4. Functional Integration.

When two applications become functionally integrated, they start to interfere with each other, and lose their separate identities as applications. As we noted above, it would perhaps be possible functionally to integrate text and graphics editing, but today this isn't very easy or practical. Text editors are big and complex pieces of software, so are graphics editors. They, and the user, have quite enough to contend with as it is.

Nevertheless, users will increasingly come to expect functional integration. When they achieve that expectation, in my view, they will no longer perceive different applications - not if the integrator has done his job properly.

Functional integration gives the appearance of a single, very powerful application covering a wide range of functionality. To differentiate a functionally integrated system from one which is simply monolithic ought to be quite hard to do from the outside.

From the inside, however, it's much easier. A functionally integrated system has performed a trick it can do again. It's extensible.

From the developer's point of view, functionally integrated systems are hard to do after the event. It's either very difficult, or very costly, to integrate functionally two applications which were not intended to be integrated. It is an architectural problem. Applications which are meant to be functionally integrated must be designed that way from the start.

2.5. Integration of User Tasks.

We have now looked at the various types of "integration" we can achieve, and it should be clear that we have really been talking about "levels" of integration. Each of the four types presupposed some ability to do the previous level.

It also seems clear that "functional integration", though it requires the others, isn't just a simple extension. It may require a more fundamental rethink of those applications which can benefit from this level of integration.

It's neither necessary, nor desirable, to integrate all applications at the functional level. Often it doesn't make sense to do that. I have no pressing need for the ability to perform calculations while browsing my filing system. I don't need my filing application to offer me a calculator. But on the other hand, I may need to perform a calculation for an (apparently) unrelated reason, WHILE I'm browsing my filing system.

That realisation, on its own, highlights the shortcomings of an integration strategy that proceeds on an ad hoc basis to bind some applications together more tightly than others. Our natural targets for functional integration - the most complete form - will be applications perceived by their designers to be "related" to a single user task. But task analysis is full of pitfalls.

A task is something that's specifiable in a time continuum. I do this, then this, then this, then I've done it. Programmers understand this sort of thing. Even though the order is sometimes arbitrary, serialising a set of activities is a programming skill. But office work is, for a lot of people, not a simple sequence, but a lot of parallel threads with switching between the threads and interrupts which start new threads and terminate old ones. There is also an information monitoring function going on as well.

Up to now, we haven't had to worry too much about this in office automation because the computer was useful for only a small part of an office worker's job - unless that job is highly structured, when by definition the threads are serial and understood.

But the user's expectation is changing. The office system of tomorrow isn't going to be just another piece of equipment on your desk - like a telephone, or a typewriter. It's actually going to supercede most if not all the other machinery. It's going to handle the telephone, the filing, most communications, provide access to data bases private and public, as well as serve the functions of a scribbling pad, a drawing board, a typewriter, a diary, a meeting room etc.

These are ambitious goals, but it's clear that integration, functional integration, of these capabilities is the reason why the customer will buy them. I can pick up my phone any time, so I'll not buy a system which stops me doing that in future. Today, I can use my calculator any time. I'm not going to get rid of it if I'm only allowed by the system which replaces it to do calculations at certain times, or in certain contexts.

The inescapable conclusion of all this is that the more comprehensive we make our office automation product functionality, the greater is the need for total integration of function. In fact, we cannot afford to think in terms of products at all, we need to devise a framework which offers functional integration as a GIVEN, but which can absorb new functionality at any time. We may be some way from offering you a telephone in your terminal, or a video conference screen, but already we have trouble making the arbitrary decisions about what will work with what. These problems will get worse when we try to implement the telephone.

3. Current Architectures.

There are a number of architectures for integrating office systems and, indeed, other applications. Now we know about integration, we can look at them and see how far they go toward our goal of total functional integration.

Before attempting such a critique, we have to bear in mind that many of these architectures were developed to encompass the functionality of applications systems that had already been written. Let's classify them immediately into POST integration and PRE integration architectures.

3.1. POST Integration.

It's much harder to do a good job of integrating existing applications, as we have already seen. HP's office systems exhibit two architectures of this type.

HPMenu provides a comprehensive ability to integrate any program running on the HP3000 into a common framework. It doesn't simply execute programs from a menu, it also provides a range of facilities to control the running of these programs, and to allow limited communication between them where this makes sense.

HPMenu can integrate programs which do not know they are being integrated. For the most useful effects, however, programs need to understand that they can communicate with each other, and in this way it's possible to go a long way along the functional integration road.

HPDesk consists of a number of little products all integrated together at the data level. HPDesk can also communicate information with HPSlate and HPWord, a trend that you can expect to see us continue for other products. HPDesk integration is much tighter than, but not as powerful as HPMenu integration. When the two are combined - HPMenu can execute HPDesk - a number of integration goals can be achieved in a very cost effective way.

Other manufacturers have taken this approach, and most POST integration architectures offer much the same facilities. There is a lot more mileage in this approach before it reaches its limits. It ceases to be cost effective only when major changes to the products are required to support it. This introduces a maintenance and configuration load on customer and supplier.

HPDesk integration succeeds, among other reasons, because the information generated by different products is stored and organised in a common way. HPMenu can only achieve this on its own with the help or hindrance of the MPE file system. This is ultimately limiting for users who are not conversant with the file system and its use by various products.

3.2. PRE Integration.

There aren't so many obvious examples of integrated systems which were conceived and implemented that way. It's not hard to see why, either. If you are going to compete with what HPMenu can do, you are going to have to rewrite every program that HPMenu can run - an impossible task.

The XEROX Star system, and the APPLE LISA (which looks similar) are the most widely known examples of systems which offer a wide range of integrated functionality. There are a number of very impressive systems emerging on personal computers - for example, the MBA CONTEXT system.

On the face of it, these systems look like big and complex programs which happen to do a lot of things others have done with separate programs. Sometimes, that's just what they are, and whether they should then be judged to be "integrated" would depend on the extent to which new functions can be incorporated.

What does seem clear, however, is that systems which try to exhibit the greatest level of functional integration seem to have a common user model. This may be a coincidence, but I don't think so. It may be worth looking at this problem from the outside, for a change, and thinking about what we'd expect an integrated office system to be like to USE.

4. The User Model.

Integrated office systems which offer a broad range of functionality need a strong user model which they offer and reinforce. Let's look at what a user model is, why it's important, and how it is determined.

4.1. What IS a User Model ?

The characteristic that isolates intelligent beings is their ability to model their environment. The model we carry in our heads is a distillation of the experiences we've had, and helps us to predict the behaviour of what we try to control.

The user's model of any system is a set of rules he has built up which explain for him how it works, and what stimuli he must give it to achieve his aims. This is as true for a motor car as for an interactive computer based office system.

Some systems are sufficiently simple for us to need very little model. If the system exhibits very few classes of behaviour, and responds to very limited stimuli, it's possible to learn them all and catalog them as related phenomena.

Once the complexity of the system gets over a certain level, we need to build mental models of its behaviour. We modify our model when it's incorrect, but we're trying to limit the amount of precious memory we have to devote to operating it. Our modeling skills as human beings get better as we grow up. Small children do not have well developed models of their environment, and need to remember more facts to survive.

4.2. Why the Model is Important.

Users of office systems are no different from any other learning being. When a user says that a system is unfriendly, or hard to learn, or hard to use, he's often making an observation on how good his model of it is, and how well that model is working.

It seems clear, therefore, that the designer of any system would do well to ensure that its users develop an effective model of it as soon as possible. But that's easier said than done.

How many of us have learned an effective subset of a complicated product, and ignore all its other features ? I know I do. This may be because I'm lazy, or it may be because I haven't succeeded in putting together a good model of it for myself. So in the absence of a model of the whole thing, I turn the system into a limited one I CAN handle.

The purpose of a model is to enable me to extrapolate from how the system behaved in one situation to how it will behave in others. So the system designer can reinforce a good mental model if he ensures that the system as a whole behaves consistently, and exhibits as few radically different types of behaviour as possible. That doesn't necessarily mean that the system cannot perform a lot of functions, but the general way it goes about them has to be consistent.

The system designer must start with a good mental model himself. His next task is to ensure that this model is reinforced by the system's behaviour - it must not act "out of character". In my experience, the most successful systems are those which do exactly what they appear to be doing - ones which have an obvious model. I think we can all accept that you can drive a car better if you have some idea what's happening mechanically. It's the same with computer systems. If you can understand to a limited extent how they work, you can use them more effectively.

This has been the bane of computer systems design for what we condescendingly call the "computer naive" user. Programmers, and most system designers are programmers, know what their systems do. They can guess at what other people's systems do, too. They have well developed models, and the models work. But it's not as easy to explain the system to a non-programmer, and some computer systems defy any non-programmer to form an effective model.

The user's mental model of a system is important to how successful he is going to be at using it. The designer's aim should be to present a system that's easy to model, and to reinforce that model in everything the system does. He does that mainly by being consistent, but he must also bear in mind the modeling tools available to his likely user population.

4.3. Existing User Models.

In some cases, the model that most users have is not the same as the one the designer had in mind. This is, as you'd expect, a source of difficulty and aggravation. If the user complains that the system is "inconsistent", the designers may not agree because they don't understand the user's model. Equally well, a system designer will sometimes be inexplicably reluctant to make an obvious enhancement which would conflict with his own model, even though it doesn't upset the model of the person suggesting the change.

Many of today's office products do not have very well developed, or very obvious mental models. Word processors, by and large, provide a metaphor for a typewriter - and include such familiar features as a carriage position indicator and tab markers. These are intended to make the word processor seem familiar to someone trained on a typewriter, but they fail to explain the way the word processor actually behaves.

A number of electronic mail systems, such as HPDesk, use a "folder model". The user picks items from contents lists and the operates on them with a reasonably consistent set of commands.

The model presented by an HPMenu configuration is very much at the mercy of the models presented by the programs that are eventually selected. HPMenu's own model is particularly common in interactive systems. Complexity of function choice is reduced by a process of refinement through a hierarchy of menus. In some cases, this will be the model operated by the selected program, and the user model will then be reinforced.

There are emerging, with the XEROX STAR and APPLE LISA, a family of systems offering a rather different user model. This model is an analogy of a desk top, with items of interest laid out on it. Items may be selected, and they will then offer a choice of allowable function. A basic set of functions (Move, Delete, Show properties etc.) will generally be available, and much of the functionality of the system is available through these generic commands.

4.4. Integrated System Models.

In a system which offers a broad range of function, encompassing all that we do in office systems today, the user's model is extremely important. We really owe it to the user to devise a model and adhere to it as far as we can in everything he's going to do. Unless and until we can do that, we aren't doing him a service.

If we simply bolt together a lot of quite individual programs, and pretend that this constitutes an integrated system, we shall find that the user is confused by that system's behaviour. He can tolerate a certain amount of inconsistency if the total number of things he has to remember is limited. But once he has to manage hundreds of potential behaviours, he will need to form some rules in order to cope.

As long as what we are giving the user is a number of quite different programs to run, it's almost better if programs are something that occur in his mental model. If he can see the product boundaries clearly, he can adjust his view of "the system" to accomodate the fact that it can behave differently in different contexts.

So we shouldn't embark on a functionally integrated system lightly, or we risk turning our users off quite quickly. If we are serious about integration, we have to pick a model and stick to it.

5. Architecture.

Our requirements for an integrated office system are that it should offer all its functionality at all times, that it should enable data and function integration, and that it should provide a consistent user model.

5.1. Object Based Architectures.

These requirements are best met by an architecture which is object based, rather than program based. An object based system consists of data structures, called OBJECTS, which are divided into CLASSES. For each class of object there is a single software unit which performs all operations on objects which are instances of the class.

Almost anything in the system can be defined as an object, and complicated objects can be broken down into simpler objects. The internal representation of an object, and how each of its functions is executed, are unknown outside the software unit which controls the class. All that is available is an interface to the object. This interface is usually documented as a set of possible MESSAGES to the object, each message consisting of the name of the function to be performed - called the message SELECTOR - and possibly some function parameters.

The user interface to an object based system consists of a mechanism for selecting an object, and a means to send it messages. One of the functions most objects will perform for you is to display a list of the functions they perform. Each function involves one message, and completes immediately. The software remembers nothing from one message to the next. Only the objects are allowed to change.

This means that, in theory, the user has all the functions of the system available to him at all times. Since complicated objects can be built from simpler ones, data integration of the sort we initially identified, is possible. Functional integration isn't an issue with an object based system, since we no longer have separate products performing common functions. What we have is different objects with common "sub-objects". Clearly, such a system exhibits functional integration by default.

5.2. Components of the Architecture.

Object based systems require a uniform way of addressing objects if they are to be easy to construct. It must be easy to build big objects out of smaller ones, in order to leverage a few simple functions. Objects are created in memory, and linked together into more complex object structures. The place where objects are kept is called the OBJECT SPACE, and it's managed by an object space manager.

An object based system requires a user interface, and it's quite consistent and feasible to treat the user interface device as an object. Since, in an object based system, only the software for the object class has any idea how the object is really implemented, the user interface object is responsible for handling the different types of user interface device likely to be encountered.

The object space is primarily a disc resident structure. It will be shared by all users, and needs to be secured against disc failure, recovered, and generally tidied up by a set of utilities. A useful utility is a general archiving facility to reduce the online disc requirement of the object space.

Where the system is one of a network of such systems, there needs to be a standard mechanism for moving objects from one space to another. Given that objects are stored in a standard way, it will be possible to copy objects from one system to another accurately. Provided that their software modules are installed on the target system, they should look and behave as they did where they originated.

Last, but not least, an object based system needs software modules which handle all the functions. Objects are divided into classes, and each class has a software module. The module is required to be able to create new instances of objects, and to perform each of the defined functions of the class. The module is therefore broken down into submodules. Each submodule is called a METHOD of the class.

5.3. Implementing the User Model.

An object based system lends itself to a particular type of user model. In the office, the model resembles a physical desk top. Objects are displayed for the user, and selected. They all respond to a small set of common commands, and individual objects may offer further operations more specific to their purpose.

The user's desk top starts out with a couple of standard items on it - an IN and OUT basket, a filing cabinet, a diary, some empty pieces of paper to write on, and some blank envelopes. He can point at these objects and have them display themselves in more detail. He can write on the pieces of paper, file them away, put them in envelopes he then labels and places in his OUT basket.

The best sort of user interface device will support a graphical representation of an object - an ICON. Icons are an aid to memory - they remind the user what the object is - and reinforce his model of the system. Ideally the terminal should have an easy way to select an object. A lot of object based systems use a "mouse" for this. The terminal should have a keyboard for text entry, and there may be standard function keys.

But the benefits of an object based approach are not restricted to graphical interfaces. Normal alphanumeric terminals can substitute textual icons for graphical ones. Pointing can be achieved by labelling or highlighting, or even by moving the text cursor. Function keys, should they be missing, can be implemented by commands. In any case, the skilled user may prefer to enter commands from the keyboard rather than by pointing to sequences of objects and labels.

Every command can be preceded by the selection of the object to which it is to apply. In the absence of a selection, the system will assume a command to the previously selected object. Text entry and editing is achieved by positioning the text cursor and typing. The underlying object is amended immediately. There will be few instances of the system requiring a specific user input in order to continue or complete an operation. The system will not be strongly "moded".

The system will maintain simultaneously the context of the user's interaction with an unlimited number of objects. The only limit is the user's capacity to keep track of the tasks he is engaged in. In this respect, the system models the way a real desk top works. Some people have lots of things on their desk. Others have only one, or nothing at all. The system doesn't dictate your work style.

Since the objects themselves have no knowledge of the implementation of the user interface object, they will interface to it in the same way, via standard messages. The user interface device is responsible for implementing pointing, selection, function keys and command input. It has no knowledge of the internal implementation of the application objects. It interfaces to them via standard messages as well.

It should be possible to add new object classes, and hence new functionality, by simply defining new software modules and data definitions. Provided these objects interface with the user interface object in the standard manner, the functionality will be immediately available on all supported terminals.

Some terminals cannot adequately display certain object types. An alphanumeric terminal may not be able to draw a picture. The implementor of the terminal object software has the discretion in these cases either to neglect to implement the display, or to translate it appropriately. Thus it could show a formal graph as a suitably labelled table of vectors. When displaying text, it may not be possible to honour the specified line width and tabulation, so the resultant display may not be quite what the printed output would be. This happens in any case for font changes and proportional spacing in all but the most sophisticated word processors.

There are some functions that the implementors of object software may not choose to implement "on-line". The system should define and support a server mechanism. A server is a background process which can execute object functions which are lengthy, or which require serial use of a resource. An example of this is the PRINT function.

5.4. User Scenario.

To get a feel for what such a system is like to use, let's consider a typical user session.

The user will execute, or have executed for him, an ordinary application program. This program embodies, or has access to, a library of object methods. All it needs are the objects.

The object space is partitioned hierarchically into a number of user subspaces. Objects may be simultaneously in more than one user space, but users will typically not be aware of this. The system needs to know which user space partition this session is to operate in. This is achieved by a signon procedure, similar to MPE signon, or the signon of HPDesk.

When the application program has located the right user space, it will find there a collection of objects the user was working on when he last signed off. This is where the system behaves slightly differently from most conventional systems. Since the only things that can remember anything are objects, and since objects are stored on disc, there's no reason why it cannot resume any session precisely from where it left off. The system provides continuity of operation, just like the desk top it models. Of course this doesn't stop you tidying your desk up before you go home, if you are that type of person.

So, assuming there were things on your desk when you went home last night, you'll still find them there in the morning. All the papers will be open at the page you were on. Your IN basket may have new things in it. Your calculator will be turned on. The graph of the sales figures, if you have a graphical terminal, will still be there.

Now you continue where you left off. Initially, the system will not know which of these objects is the one you're interested in. So you'll have to tell it. If you have a pointing device, you can point to it, and it will then highlight itself to confirm that it's the current object. If you don't have a pointing device, you can type the object's name, or have the system serially highlight all the objects until it gets to the right one. If it had been partially obscured by something lying on top of it (a problem with desktops everywhere) it will move itself to the top of the heap.

You can now perform a number of standard operations on this current object. If it is a complicated thing, like a folder or a document, you quite probably don't want it displayed all the time. So one of the things you can do to most objects is to OPEN it. This will cause it to display itself in a new screen window. Alternatively, if you aren't interested in the object for the moment, you can CLOSE it. This will shut down its window, and if there are no other windows on the object, will make it disappear entirely. All objects you have must have initially been obtained from some other object, so you'll presumably be able to get it back.

You can always move the object display around the screen, perhaps so that you can better see other objects, or just to get them out of the way. If you move an object deliberately to the space occupied by another one, the system will try to interpret this in a sensible way. If you move a document to a folder, that's equivalent to filing it. If you move an envelope to the OUT basket, that mails it. If you move an object to a "printer object", that prints it. In particular, if you move an object to your wastebin, that's almost the same as deleting it. Until the end of your session, you will always be able to recover it from the wastebin.

On a graphical terminal with a pointing device, these effects are all achieved by pointing. On other terminals, it can be achieved with commands like "MOVE TO WASTEBIN". Subject to loss of display fidelity in obvious cases, it ought to be possible to operate the system from any terminal.

When an object has been OPEN'd, it will typically offer some further functions beyond the standard OPEN, CLOSE, MOVE, COPY etc. Documents, for example, will let you turn over their pages, and type on or over them. IN Baskets will show you lists of messages you have received, and you can read one by pointing to it and opening it.

If you want to create a new object, you do this usually by copying from an existing one of the same type. The existing object might be blank, or it might be a standard, "boiler plate" for its type or a complete one. So to start a new document, you find an old one, select it, and issue the COPY command. You now have a new document, with a display window on it. You can continue doing the same things to this one as you could to the old one. The system will provide you with a sort of "stationery cupboard" consisting of blank objects of the types it has software for.

There will be cases where you want to access software which isn't functionally integrated into the system in the fullest sense. There are two ways you can do this. It's fairly trivial to implement a "program" object which accepts a RUN command, or for that matter an "operating system" object which accepts any operating system command. When you run the program, it will take over your terminal, and that will be the end of that until it terminates.

Alternatively, the program you want to run may be sufficiently tame and device independent that it can use standard input and output devices. The implementation of such a program object would create a window for it to read and write to. The advantage of this is that the user interface could treat this as a standard text window similar to one you typed yourself. Then it would be possible to move the output of the program into your documents and mail messages. This can be useful if your program was, say, performing a data base extraction.

In the longer term, we'd like to integrate as many programs functionally as we reasonably can. This will sometimes mean writing object based software which accesses conventional software but interfaces normally to the remainder of the object based world. For example, suppose we implement a spreadsheet object. This will behave to the rest of the world like a table. Tables are things you can give to graph objects and have them draw for you. But a table is also the sort of thing you can get by enquiry from a data base.

So you can envisage a situation in which the system implements a number of standard or user configurable "data base" objects which, when they display themselves, do so as tables. You could even update the data base by amending the table, if the object allowed you to do so. But perhaps more practically, you get, almost for nothing, the ability to plot a graph from data base data. This is possible because, in an object based system, the only software which has any business with an object's implementation is the software which defines the object class. Thus, all the graph object knows about the tables it plots is the information it can get from them. It has no idea how, or if, the table is stored in the object data structure. Thus a graph can plot a set of data base figures just as well as it can plot a spreadsheet, or figures actually typed in by the user to a real "table" object.

This should give you some feel for what is possible in a functionally integrated system using an object based architecture. Clearly, these effects cannot all be achieved overnight. We can go a long way with a basic set of objects and loosely integrated program objects to cover other functions. Gradually, the system will evolve into a more completely object based system. Conventional software can be interfaced by implementing objects for the purpose.

Of the possible functionality that can be offered, we cannot expect more than a basic set of applications from the supplier. We would expect that, in the long run, users and customers should be able to add their own objects or reimplement the supplied ones. Object based systems are not essentially more difficult to write, but they do require a different approach. By comparison with conventional software, it is much more highly leveraged. New object types can be implemented using existing ones, or by defining subclasses of existing ones. In particular, a standard user interface facility is offered that other software would do well to employ.

6. Conclusion.

We have looked at what it means to integrate office applications, the types of integration there are, and why we do it. We identified the two main architectural approaches to integration.

We also looked at how systems are modeled by their users, and why that model is critical to user acceptance and effectiveness. We considered the particular problems of modeling and using a complex integrated office system.

Finally, we explored an architecture which meets the users expectation for functional integration, and his need for an acceptable model.

USER TRAINING IN OFFICE SYSTEMS

Jay Young

Office Productivity Division

INTRODUCTION

Let me start by defining what I mean by User Training. I do not mean purely product training, that is, how to use the Word Processor or Financial package. I mean a much more comprehensive approach, covering an introduction to the new concepts and specific products, an awareness of how they can increase their users' productivity and effectiveness, training in how to use the products and the period of consolidation, practice and further training that follows every implementation.

This definition is product specific - that is, it does not address the generic issues. These include how people feel about the introduction of office systems in general, their fears about their job security, their future prospects, their working conditions and health. These are all issues that an effective office systems implementation and training program should address.

However, today, I want to confine myself to the product specific issues in training office users.

What I intend to do in this presentation is to develop a 4 step approach to training users on a new office product. Office vendors should provide training programs which cover all the aspects discussed. Such a program will start shortly before the product is actually installed, and may carry on for several months if there are a large number of users involved.

WHAT SHOULD THE TRAINING PROGRAM ACHIEVE?

It is important to establish the objectives of the training program. As I have already said, I do not believe that it is enough for a training program to confine itself to teaching simply HOW to use the product. There are a number of other objectives that we would like to achieve.

First, we want people to be able to use the product confidently. They must understand how to use the product to carry out the particular applications that they have. A training course that does not teach people the features that they will need to use, however esoteric the application, will not achieve this objective.

Secondly, we want people to be, if not positively enthusiastic about the product, at least comfortable using it. We want them to understand how the product is going to be helpful to them in their work, and to be motivated to use it.

Thirdly, we want people to make appropriate use of the product. The training program should give them the opportunity of looking at their department's or group's work and finding ways of using the new product to improve on current methods. If an office vendor is selling productivity tools, they should make every effort to help their customers use them productively and help them to find appropriate applications.

Fourthly, we want managers to have the necessary skills to run a department where the new product is being used. They must understand the impact that the new product may have on their department and be able to handle any changes to work loads and methods and working relationships.

Lastly, we want to develop an effective post-training support service. Users of a new product will inevitably run into problems, however good their training course. They require access to some person or persons who can help them to solve these problems. A general forum for communicating ideas, applications and issues between users would also be beneficial.

Now that the objectives have been defined, we need to consider some of the factors that may influence our ability to achieve them.

IMPORTANT CONSIDERATIONS

Office users are a heterogeneous group. Everyone in an organisation is potentially affected by the introduction of office systems and training programs must be able to address the needs of each individual or group of individuals.

What are the characteristics of these groups?

First of all, they are at all levels of an organisation, from Managing Director downwards. The training program must be able to offer training and support to all these people, and in a way that is appropriate to their position, daily schedule, and computer understanding.

Secondly, there will be a widely ranging level of interest in the new product, from enthusiastic to perhaps hostile. Our program must try to seek out people's feelings and to motivate them both to learn and subsequently to use the product.

Next, they will have varied knowledge of computers. Some potential users may have their own Personal Computers at home, others will be completely new to a computer. The program must be able to give confidence to the "new" users and also satisfy the greater desire for knowledge (and perhaps recognition) of the more experienced users.

The preferred method of giving product training may also vary between groups. There will be groups of users (such as the existing computer users) who will be able to follow self-paced training programs, perhaps computer based; others may prefer a classroom course which will enable them to concentrate on learning the product, thus becoming proficient in a shorter time. Others may require individual coaching sessions. The training needs of senior managers must be considered. Product training programs must be able to cater for all the varying requirements.

Finally, a wide range of applications will exist. There will be many different uses for the product. There will also be uses that may not be immediately obvious to new users. Our program must provide help to users in analysing their work and looking for ways in which the new product can improve their efficiency and enjoyment of their work.

To summarise, our training program, while achieving its objectives of user confidence and motivation, appropriate product usage and effective support, must also be able to address the problems presented by the positions, the degree of interest, the level of computer knowledge and differing applications of the users.

How can we achieve this? The rest of the paper concentrates on looking at a suggested 4 step training program which sets out to do just this.

STEP 1 - INTRODUCTION

The first step in the program is to introduce the users to the new product that they will be using.

This introduction serves three main purposes:

It introduces users to the concepts behind the system. For someone who has no idea of what, for example, an Electronic Mailing system is or of what it can do, it is important that they understand something about its purpose and capabilities before setting out to learn HOW to use it. The user will need to know:

what does the product do?
what can I use it for?
why should I use it?
what will it do for me?
what does it look like?

The introduction also provides an opportunity of discussing feelings and anxieties, and of motivating the users. As I have already discussed, not all users will be enthusiastic and eager to use the new system. Perhaps they are being asked to switch from a word processing system they have used for some time - they may not feel inclined to learn yet another system. Perhaps they simply prefer to continue using the existing systems, automated or not.

During the introduction, examples of the various uses of the products can be given. Specific advantages of the product over previous methods can be highlighted and these should be geared to each group being addressed. Some indication should be given of the learning time involved so that expectations can be correctly set.

It is often useful for prospective users of a new product to hear of the experiences of existing users. This allows them to ask questions of someone with whom they can identify better than they can with the vendor's representatives. If a pilot study has been carried out, a member of the pilot group could do this.

Finally, the introduction can include a demonstration of the product (this could be done 'live' or on a videotape). This enables everyone present to get an idea of what the product looks like, how easy it appears to be to use and how to carry out some simple functions.

The introduction need only be a short session which should be given to all prospective users. Its length will depend on the complexity and novelty of the product being introduced, but it should not be more than one to one and a half hours.

This session should ideally be given to homogeneous groups, that is, all the people in a particular work-group or department should attend together. This enables examples and demonstrations to be tailored more specifically for the audience. It will also ensure that the same information has been given to all the members of each department which will help them when they come to discuss their own application of the product.

Some introductory documentation covering the points made in the introduction and also an explanation of the features of the product can be distributed at this time.

So, the objectives of this session are to explain the concepts and some basic features, to give a demonstration of them and to motivate the users both to learn and use the new product.

By giving such an introduction, we generate interest in and perhaps enthusiasm for the product and provide some basic information for users to carry forward into the next stage of the program.

STEP 2 - ANALYSIS

The next step in the program is to give users an opportunity to think about how the product applies to their particular work group.

The purpose of this is to encourage each work group to look at the work they do and find effective ways of using the new product to help them. Not all their activities will necessarily be better done using it. An understanding of where the productivity gains are to be made needs to be reached.

Vendor representatives should help with these group discussions, acting as a facilitator rather than directing the discussion. They can also provide additional information about the product if required.

I believe that it is important that as much help as possible is given to the customer to understand where a product can be of most benefit and to realise the productivity gains that have been promoted.

A number of benefits are to be gained from this analysis stage.

Users will acquire further knowledge and understanding of the product's functions. They will reach a greater understanding of how it can be used to improve the quality and effectiveness of what they do.

They will be able to discuss as a group the work they do, the current systems they use and how they might be able to do their work better, with or without the help of the new product.

With a product like Electronic Mail, this is absolutely essential, especially for managers and their secretaries who will need to work out how they are going to use the product most efficiently to deal with day to day events and correspondence that have always been handled by the secretary in the past.

Finally, by the end of the discussion, each group should be able to define their own training needs. This is possible because they have identified the uses they have for the product and therefore the features they need to learn.

These will vary from group to group. For example, with a Word Processor, the Finance group's main application may be to type columns and rows of figures and use calculations. They may have no need to learn about using Headings and Footings in a document although this might be an important feature for the Legal Department users.

This definition of training requirements is extremely helpful in outlining the formal training courses which are the next step in our program.

In summary, the objectives of the analysis step are to develop with each work group a list of applications that they have for the new product. In addition, each group should identify their individual training requirements.

By going through this analysis, we hope to continue to generate enthusiasm through an understanding of the product's specific value to individual work groups, and to ensure appropriate and optimal use of the product throughout the user group. We improve the prospects for a successful formal training program by identifying individual applications and therefore training requirements. The work group approach ensures that all prospective users, regardless of position, can further their understanding of the product and can contribute to planning its use and effectiveness in their and their colleagues' jobs.

STEP 3 - FORMAL TRAINING

Step 3 is the formal training, the aim of which is to provide the necessary level of basic training to enable all the users to get started confidently with the new system.

The specific objective will be to train each user to the level that they require in their job. Training objectives will have been set for each individual and/or work group during the previous phase. This should enable the formal training program to be planned very precisely according to these predetermined requirements.

The method of delivering training may well vary from group to group and it is important that a variety of approaches are offered. Classroom courses, self paced training, perhaps computer based, and individual coaching may all be required.

CLASSROOM COURSES

Classroom courses have proved to be the most effective way of carrying out certain types of training, for example, word processing where it is important that the operators learn quickly and thoroughly.

These courses should be available either at a vendor Training Center, or on the customer's site. In the latter case, the course can be adapted to suit the specific needs of the class.

Although it is probably best for the vendor's experts to give the initial training classes, some companies will have training staff who can take on this responsibility as the implementation of the new product progresses.

SELF-PACED TRAINING COURSES

Self-paced training is valuable for those who can schedule their own time to take the training and who do not need the additional instructor explanation and support that is available in a classroom. It is also useful for those who want to learn additional features of a product, after learning the basic ones in a classroom course.

To be effective, self-paced training requires the availability of an expert who can answer questions and clarify any particular problems.

If self-paced training is to be used, it is probably best carried out in a Learning Center. The value of a Learning Center is in providing a quiet environment for learning and a supervisor on hand to help solve problems and give additional advice and information if required. However, the Learning Center is likely to be an inappropriate method early in the implementation of a new system, when there are no company experts to act as supervisors.

MANAGEMENT COURSES

Management training falls into two categories.

The first is formal training on how to use the product. The second is the training in the management skills required to make the most effective use of the new product in the individual manager's department.

A recent survey of member organisations by the International Word Processing Association showed that whereas 70% of the companies surveyed provided training for their word processing operators, only 25% provided any training program for principals, and for senior management, only 8%.

Of the management training programs that were provided, only formal skills were covered (and the most common skill taught was Dictation!).

To achieve its objectives, our training program should provide management training which covers:

a review of the product
applications discussion
basic product training
effective ways of using the product
managing an "automated" department

Such a program will ensure that all levels of management receive training appropriate to their needs. It should also generate additional commitment for the new product from management, a very important factor in the product's success. Finally, it will enable them to manage their group under the new system, being aware of the possibility that existing jobs may change, the way the group is orgnaized may need to be changed, and so on.

SUPPORT SERVICES

One of our objectives for the training program as a whole is to ensure the development of an effective support service. The training courses should explain to the users what arrangements have been made for support after the training course. Problem solving should be covered as part of the course, together with the use of the documentation for self-help.

So to sum up this formal training stage, the objectives are to provide user training which will enable each user to reach a level at which they can use the product for their specific applications, and to provide management training which will enable managers to cope with the impact the new system will have on their departments.

By doing this we hope to achieve our aims of confident users at all levels of the organisation, we cater for the needs of both 'new' and 'experienced' computer users, and we enable management to supervise effectively the use of the new product in their departments.

STEP 4 - FOLLOW UP

The final step in our training program is the period of consolidation with the product being used in the offices.

A training program should not stop as soon as the users have been through their formal training. It is very important that during a follow up period of say 8 weeks, users are given support and assistance in their early use of the new product.

The main support service offered will probably be a HELPLINE for users to call up for help. But users should also be encouraged to try to solve their own problems with the help of Reference Guides, or asking each other.

The support staff can also arrange meetings of users to exchange ideas, problems and workarounds. The vendor's instructors could take part in some of these meetings to help with specific requests.

It is likely that during this period, further training requirements will be identified - applications that perhaps were not thought of in the earlier stages, correction of misunderstandings, or simply a readiness to move on to new and more advanced features.

Arrangement should be made to meet these further training needs, through additional training classes, self-study, user meetings/seminars or any other method that seems appropriate.

The objectives of this final step are to enable the users to work with the system, consolidating the knowledge they have gained in training, and learning to use the support service. It also provides a time for testing the effectiveness of the support service.

By including a period of follow-up in our training program, we continue to build the users' confidence and even enjoyment of the new product by maintaining interest and support in their use of it. We can see the impact of the new product throughout the organisation and ensure that management is able to handle any necessary changes. We review the usage of the product in the different departments, ensuring the smooth running of existing applications and helping to identify new ones. We also test the support services that have been set up.

CONCLUSION

So the training program we have developed is a 4 step plan, covering:

INTRODUCTION
APPLICATION ANALYSIS
FORMAL TRAINING
FOLLOW UP

We hope to have achieved our 5 objectives, namely:

Confident users
Enthusiastic users
Appropriate usage
Informed management
Effective support service

We should also have covered the differing needs presented by a heterogeneous group from all levels of the organisation, with mixed feelings about the introduction of the new product, some of whom may know nothing about computers, and who have many different applications for the product.

Finally, we have given management of all levels some additional help in running departments where the new product is being used.

I hope that this presentation will provoke some thoughts about the training programs that are offered for the implementation of office systems, and will give you some guidelines for evaluating them.

THE FCS-EPS DECISION SUPPORT SYSTEM - BUSINESS PLANNING ON HP3000 COMPUTERS

FCS-EPS is a specialised decision support system generator for financial and resource planning, analysis, and reporting. It can be easily learned and used by non-programmers to develop models in a minimum of time. In spite of its ease of use, FCS-EPS has a wide range of capabilities which go beyond the basic modelling tasks to ensure that the user will never outgrow the system as his expertise and the complexity of his applications increase. Users have found that the system is capable of handling any design concept their requirements call for, including completely black-boxing or front-ending applications so that even untrained users can easily run models written by others by simply answering questions.

Model building in FCS-EPS closely follows the business analyst's natural thought process. Using the English-style modelling language, he enters the variables and the applicable input data or calculations for each.

In order to illustrate the concept of how FCS-EPS works we have created a ficticious organisation, K.F. Ltd., who are committed to the production of annual budgets as the basis of their financial control system. The demonstration will lead us step by step through the creation of a "budgeting model" and show how it may be used to evaluate and control the financial future of K.F. Ltd.

K.F. Ltd.

K.F. Ltd. require a system to reduce the work associated with the compilation, consolidation, analysis and presentation of quarterly budgets.

For the purpose of demonstration the company has been simplified to have 5 factories in 2 divisions. The Engineering Division has factories in Leeds, Croydon and Bristol while the Components Division has factories in Manchester and Newcastle. However, the principles can be extended to any number of factories in any number of divisions. Similarly any other basis for aggregation would be equally valid, for example, Product, Cost Centre, Department.

The first part of our demonstration shows how a simple budget may be prepared for the Manchester Factory. The LOGIC is created, the DATA input, the REPORTS produced and SENSITIVITY ANALYSES performed.

The second part of our demonstration shows how the budgets for each factory may be consolidated after each has undergone a similar process. The total budget is evaluated by producing comparison reports across divisions and factories.

K.F. Ltd - The Budgeting Process.

The nature of the business that K.F. Ltd. find themselves in, i.e. manufacturing to demand, dictates that all of their plans are dependant upon their Sales Forecast.

Thus, in considering how best to create a typical factory budget, K.F. begin with a projected volume of sales, of the product of such a factory, by planning period.

Then, since it is planned that all of the projected sales be satisfied by production in that period, the variable costs of production are directly derived and hence the factory contribution.

Next, the factory fixed cost overheads are estimated and phased over the entire period of the plan, in order to arrive at an overall factory profitability.

Finally, the individual factory budgets are consolidated to create both the divisional and company profit statements.

The Way It's Done Now - Manually.

In addition to the budgeting process previously outlined, it is important to understand that recalculation of all or some of the budget information may be required when the forecasts are not acceptable. This situation may occur at factory, divisional or company level and may result in many changes being made throughout the organisation.

When manual methods are being employed, as at K.F., the amount of "number crunching" required when computing the initial budget often takes so much time that little opportunity is allowed for detailed analysis of the projections. Alternatively, time is made by commencing the budget calculations at an earlier date, however, this often has an impact on the accuracy of the forecasts being made.

Inevitably in such circumstances, approximations are made, alternatives are not assessed, errors occur and mid year budget revisions are not made any easier.

Faced with such a situation, K.F. Ltd., decided to employ a computer based Decision Support System, FCS-EPS, in order to allow them the opportunity to analyse and improve forecasts and begin to control their business more exactly.

Entering Calculation Rules.

Since the budget appropriate to K.F. Ltd., is to be based upon "reality", then the model to be constructed will be based upon the financial framework in which the company's activities of procurement, production and selling are represented in financial terms. This representation will thus be analogous to accounting statements and will incorporate such items as revenues, costs, contribution and profit.

Thus, the model will be defined from quantifiable factors and the relationships between thse factors, see Figure 1. The relationships in the model are often referred to as the 'logic' of the model.

The LOGIC command has the prompt: +. In response to this, the user tells FCS-EPS what factors are to go in each row of the underlying model matrix.

e.g. 12 'Number of Weeks'.

All rows must have a number. The row name where appropriate must be in single quotes.

A row such as the one above would be considered to be an input row, i.e. it requires data.

Where a row is to be calculated the definition of that row is extended to incorporate this definition,

e.g. 18 'Sales Revenue' = 'Sale Price' * 'Volume'

or when utilising the row numbers,

34 'MATERIAL' = 14 * 24

This second method providing a faster method of input but less clear documentation.

The LIST NAMES command allows the user to enjoy the best of both worlds, since it produces a documented model from the row number defined input.

Calculated rows employ the usual computer operators

+ ADD, - SUBTRACT, /DIVIDE, * MULTIPLY

as well as over 100 additional operators and functions, such as

AT,	multiplies by a percentage
%,	calculates a percentage of
SUM,	adds an inclusive range of rows
LAG,	references previous columns
INTEREST,	computes the interest payable or receivable on a cash balance.

The systems line by line error checking, for syntax and compilation, enables errors to be pointed out immediately upon input and the maintenance of a compiled model, which may be altered line by line without recompiling the whole.

It should be pointed out that non-procedural calculations, for example where the use of forward references in the logic are deliberately used, are allowed by the system and as such are not trapped as syntax errors,

```
e.g.   50  'Gross Profit
       60  '% Commission'
       70  'Commission'   =    80  AT 60
       80  'Net Profit'   =    50  -  70
```

Lastly, having prepared a logic file in this way the SAVE LOGIC command is used to keep a permanent copy of the model for future use.

```
System>SET COLUMNS 5                           Specify the number of columns
                                               (4 Quarters and Annual Total)
System>LOGIC

+12 'Number of Weeks'                          Define the variables in the
+14 'Volume'                                   model and the relationships
+16 'Sales Price'                              between them.
+18 'Sales Revenue' = 'Sales Price' * 'Volume'
+22 'CPU Labour'
+24 'CPU Material'
+26 'CPU Carriage'
+28 'CPU Energy'
+32 'Labour' = 'Volume' * 'CPU Labour'
+34 'Material' = 14 * 24
+36 'Carriage' = 14 * 26
+38 'Energy' = 14 * 28
+42 'Contribution' = 18 - 32 SUM 38
+46 'Prodn Overhead'
+48 'Sales & Admin'
+50 'R&D'
+54 'Net Profit' = 42 - ( 46 SUM 50 )
+64 'Rtrn on Sales %' = 'Net profit' % 'Sales Revenue'
+

+LIST 42-54                                    Examine any part or all of
+42 'Contribution' = 18 - 32 SUM 38            the model.
+46 'Prodn Overhead'
+48 'Sales & Admin'
+50 'R&D'
+54 'Net Profit' = 42 - ( 46 SUM 50 )
+
                                               Self-documenting feature.
+LIST NAMES 42
42 'Contribution' = 'Sales Revenue' - 'Labour' SUM 'Energy'
+

+REPLACE 'CPU'Cost/U'
22 'Cost/U Labour'
24 'Cost/U Material'
26 'Cost/U Carriage'
28 'Cost/U Energy'
32 'Labour' = 'Volume' * 'Cost/U Labour'
+

+END
System>SAVE LOGIC                              Save the Logic for future use.
Logic file:/ABUDL1
System>
```

FIGURE 1

Entering Data.

In order that the required information may be derived from the model, e.g. the Factory Net Profit, clearly the estimates of factors such as volume of sales and selling price need to be supplied to the model.

This data entry may be done by answering DATA at the SYSTEM prompt, Figure 2, and then proceeding with line by line data entry of the "Free Form" Row Number, Data Code, Numeric Values,

e.g. 12, U, 13, 12, 11, 12

to represent 13 Units in column 1, 12 in column 2, 11 in column 3 and 12 in column 4, of row 12.

The following data codes describe the order of the numbers

U = Units
K = Thousands
M = Millions
C = Hundredths

In addition, growth patterns can be associated with data,

e.g. 14, I, 1000, 10

to represent incremental growth of 10 units per period from a base of 1000 in the first period, thus the four values 1000, 1010, 1020, 1030 are generated.

Such growth patterns include

I = Incremental
A = Arithmetic
G = Geometric

A further requirement satisfied by these data codes in PHASING. Thus on entry an annual amount, say, may be allocated to individual periods according to some pre-determined pattern.

e.g. 46, D, 80000, 12

to represent a total of 80000 in row 46 phased according to the data in row 12.

Alternative data entry methods include the ability to interface with company files such as an IMAGE database, full screen data editor from user defined screens and interactive data entry from user defined prompting.

Naturally, all data entry methods allow the input data to be validated before acceptace and subsequent use.

```
System>DATA

*FOR 1-4                                           Input data for the 4 quarters

*12,U,13,12,11,12
*LIST 12
Row    12                                          Data is input in a convenient
        13.00      12.00      11.00      12.00     form and expanded on input

*14,I,1000,10
*LIST 14
Row    14
      1000.00    1010.00    1020.00    1030.00

*16,G,150,3
*LIST 16
Row    16
       150.00     154.50     159.14     163.91

*22,U,*30
*LIST NAMES 22
Row    22    Cost/U Labour
        30.00      30.00      30.00      30.00

*24,U,*45
*26,U,*10
*28,U,*5
*

*46,D,80000,12
*LIST NAMES 46
Row    46    Prodn Overhead                        Annual production overheads
     21666.67   20000.00   18333.33   20000.00     are phased

*48,D,60000,12
*50,D,40000,12
*END
System>SAVE DATA
Data file:/ABUDD1
System>
```

FIGURE 2

Calculation.

After the logic and the data have been entered using the LOGIC and DATA commands or retrieved from their files by the commands LOGIC USING and DATA USING the model may be calculated by the CALCULATE command.

This command causes the calculation definitions to be performed and the results to be created internally to the system ready for reporting.

When a model is under development it is often useful to both limit the range of the model to be calculated and to trace the results of calculations, both alternatives are possible within the CALCULATE command.

Quick Report.

Following the use of CALCULATE two levels of reporting are available to the user. The first of these the "quick report" or "working report" may be retrieved by the LIST command.

This command allows selective output from the internal results to be produced, the selection criterior being the columns and rows of the model that are required.

The report gives immediate access to any results but in a system defined layout, Figure 3. This layout is entirely general and consists of all rows selected that contain data. These rows are printed in the selected sequence and are annotated with both their row number and the first 16 characters of their row name. The numeric values are shown with two decimal places and where appropriate are scaled, K - Thousands, in order to fit an eight character width field.

This report is designed to facilitate rapid model development since it allows the calculations to be easily monitored and amended where appropriate.

The working report may be further ammended by the addition of an appropriate title and column headings, see Figure 4.

```
System>CALCULATE                               Complete the data matrix
System>

System>LIST                                    Working report featuring
Columns?1-4                                    selective output and
Rows?1-64                                      automatic scaling

                            1       2       3       4

  12 Number of weeks    13.00   12.00   11.00   12.00
  14 Volume           1000.00 1010.00 1020.00 1030.00
  16 Sales Price       150.00  154.50  159.14  163.91
  18 Sales Revenue     150.00  156.05  162.32  168.83 K
  22 Cost/U Labour      30.00   30.00   30.00   30.00
  24 Cost/U Material    45.00   45.00   45.00   45.00
  26 Cost/U Carriage    10.00   10.00   10.00   10.00
  28 Cost/U Energy       5.00    5.00    5.00    5.00
  32 Labour             30.00   30.30   30.60   30.90 K
  34 Material           45.00   45.45   45.90   46.35 K
  36 Carriage           10.00   10.10   10.20   10.30 K
  38 Energy           5000.00 5050.00 5100.00 5150.00
  42 Contribution       60.00   65.15   70.52   76.13 K
  46 Prodn Overhead     21.67   20.00   18.33   20.00 K
  48 Sales & Admin      16.25   15.00   13.75   15.00 K
  50 R&D                10.83   10.00    9.17   10.00 K
  54 Net Profit         11.25   20.15   29.27   31.13 K
  64 Rtrn on Sales %     7.50   12.91   18.03   18.44


System>LOGIC                                   Amend the logic to calculate
+58 COLUMN 5 = 1 SUM 4                         annual totals
+END
System>SAVE LOGIC /ABUDL1
System>CALCULATE                               Calculate amended logic
System>LIST
Columns?1-5
Rows?18,42-64

                            1       2       3       4       5

  18 Sales Revenue     150.00  156.00  162.32  168.83  637.19 K
  42 Contribution       60.00   65.15   70.52   76.13  271.79 K
  46 Prodn Overhead     21.67   20.00   18.33   20.00   80.00 K
  48 Sales & Admin      16.25   15.00   13.75   15.00   60.00 K
  50 R&D                10.83   10.00    9.17   10.00   40.00 K
  54 Net Profit         11.25   20.15   29.27   31.13   91.79 K
  64 Rtrn on Sales %     7.50   12.91   18.03   18.44   14.41

System>
```

FIGURE 3

```
System>TITLE                                   Define single line title
Title:MANCHESTER FACTORY REPORT 1981
System>


System>HEADINGS                                Define single line column
Option:HELP                                    headings
    MO - MONTHLY
    ST - STANDARD                              'HELP' is available at any
    NU - NUMERIC                               EPS prompt
Option:ST
Column,heading
:1,QTR1
:2,QTR2
:3,QTR3
:4,QTR4
:5,YEAR
:END
Option:END
System>LIST
Columns?1-5
Rows?18,42-64
                MANCHESTER FACTORY REPORT 1981

                         QTR1    QTR2    QTR3    QTR4    YEAR

  18 Sales Revenue     150.00  156.00  162.32  168.83  637.19 K
  42 Contribution       60.00   65.15   70.52   76.13  271.79 K
  46 Prodn Overhead     21.67   20.00   18.33   20.00   80.00 K
  48 Sales & Admin      16.25   15.00   13.75   15.00   60.00 K
  50 R&D                10.83   10.00    9.17   10.00   40.00 K
  54 Net Profit         11.25   20.15   29.27   31.13   91.79 K
  64 Rtrn on Sales %     7.50   12.91   18.03   18.44   14.41

System>
```

FIGURE 4

Specifying The Report.

In order to produce "formal" reports, that is reports in the accepted standard of the organisation, it is necessary to use the REPORT command.

This command will prompt.the user for a series of report specifications that define the content of the required report.

These specifications allow the user to not only reference rows of information already defined within the model but also to define additional information such as page headings, paragraph headings, footnotes as well as additional subtotals etc.

The default report specifications are extensive and thus a single specification may be sufficient to define a full report, however, it is more common that many of the defaults are adjusted throughout the report.

The report specifications are fairly easy to interpret, thus WIDTH 10 SUPROW specifies that numeric fields are 10 characters wide and not the default 8 and also that the printing of row numbers is to be suppressed.

Rows 18 or the shorthand R18, will include row 18 in the report including both the row name and the associated numeric values.

Text 3 'Less Variable Costs', will cause the specified text to be included starting at the 3rd character from the left.

UDATA, causes the previous row of data to be underlined with a '-' character.

SKIP, leaves a blank row in the report.

AFTER '%', causes a % sign to be printed after all subsequent numbers.

Such specifications may be stored on a file for repeated use and any number of such files may be saved.

In total there are over 60 such specifications, designed to allow the user total flexibility in the production of reports. Thus enabling organisation standards to be maintained.

Ultimately, the report so defined may be produced from the DISPLAY command, see Figure 5.

Naturally, the report may be produced on a printer as well as on the screen or alternatively saved to a file for later printing or for passing to another computer system.

```
System>REPORT USING /ABUDR1                      Recall previously defined
                                                 report specifications
System>DISPLAY
                                                 Print the report at the
                                                 terminal
16:42:37 on 06/12/80

                    MANCHESTER FACTORY REPORT 1981

                      QTR1      QTR2      QTR3      QTR4      YEAR

    Sales Revenue    150000    156045    162318    168826    637189
  less Variable Costs
    Labour            30000     30300     30600     30900    121800
    Material          45000     45450     45900     46350    182700
    Carriage          10000     10100     10200     10300     40600
    Energy             5000      5050      5100      5150     20300
                      ----      ----      ----      ----      -----
    Contribution      60000     65145     70518     76126    271789
  less Fixed Costs
    Prodn Overhead    21667     20000     18333     20000     80000
    Sales & Admin     16250     15000     13750     15000     60000
    R&D               10833     10000      9167     10000     40000
                      -----     -----     -----     -----     -----
    Net Profit        11250     20145     29268     31126     91789
                      =====     =====     =====     =====     =====

    Rtrn on Sales %      8%       13%       18%       18%       14%

----------------------------------------------------------------------
System>

System>REPORT                                    List the report specifications
:LIST
   10 WIDTH 10 SUPROW
   20 ROWS 18
   30 TEXT 3'less Variable Costs'
   40 ROWS 32-38 UDATA
   50 ROWS 42
   60 TEXT 3'less Fixed Costs'
   70 ROWS 46-50 UDATA
   80 ROWS 54 UDATA'=' SKIP
   90 AFTER'%' ROWS 64 SKIP UPDASH
:
```

FIGURE 5

What If?

Having established a working model for the production of factory annual budgets, we are now in a position to use the model to evaluate the effect on Profitability of alternative courses of action, and changing economic climate, thus overcoming a known shortfall of the manual system. A number of commands are available to allow such analyses to be performed.

The SENSITIVITY command enables us to temporarily effect our base data by the selective addition or subtraction of a percentage or absolute amount across a range of periods.

Thus, for example, we can quickly examine the combined effect of a 5% wage increase from quarter 2, a price increase of 4% from quarter 1 and a loss of sales volume of 1% also for quarter 1, on the factory profitability.

At the end of such an analysis the data resets to the base figures in order that additional analyses may be examined on an equitable basis.

In the circumstance where the effects of a number of SENSITIVITY commands are to be compounded, then this is easily achieved by the SENSITIVITY LEAVE command.

TARGET is a powerful 'what if' command which enables one to work back from a target in order to determine the value of a variable required to meet such a target, for example what sales volume is required in order to give an annual return on sales of 20%. See figure 6.

```
System>SENSITIVITY                                  What if labour costs increase
Row,  Change?                                       by 5% from quarter 2
:22,5 FOR 2. TO 4.                                  forcing a price increase of 4%
:16,4                                               causing a 1% fall in sales?
:14,-1
:END
Columns to list?1-5
Rows to list?14-22,32,42,54,64

                     MANCHESTER FACTORY REPORT 1981

                      QTR1     QTR2     QTR3     QTR4     YEAR

14 Volume            990.00   999.90  1009.80  1019.70  4019.40
16 Sales Price       156.00   160.68   165.50   170.47   652.65
18 Sales Revenue     154.44   160.66   167.12   173.82   656.05 K
22 Cost/U Labour      30.00    31.50    31.50    31.50   124.50
32 Labour             29.70    31.50    31.81    32.12   125.13 K
42 Contribution       65.34    69.17    74.73    80.52   289.76 K
54 Net Profit         16.59    24.17    33.48    35.52   109.76 K
64 Rtrn on Sales %    10.74    15.05    20.03    20.44    16.73



System>TARGET ALL                                   What would we need to sell
Target row,column,value?64,5,20                     to achieve an annual return
Variable row,column,best estimate?14                of 20%

     Volume              Iteration      Rtrn on Sales %

      4060.00                1             14.41
      4864.88                2             19.08
      5023.64                3             19.82
      5061.13                4             19.99

System>
```

FIGURE 6

Divisional and Company Consolidation.

We will now recall that the Manchester Factory is one entity within the K.F. Ltd. company structure. The entire structure previously defined consists of 5 factories viz. Manchester, Newcastle, Leeds, Croydon and Bristol and 2 Divisions viz. Components and Engineering comprising Manchester, Newcastle, and Leeds, Croydon, Bristol respectively.

The Hierarchical Structure of FCS-EPS allows a complete definition of KF Ltd., and the ability to maintain consistent data for each factory and division. The data is referenced by Section Number:

1 - Manchester
2 - Newcastle
3 - Leeds
4 - Croydon
5 - Bristol

7 - Components
8 - Engineering
9 - KF Ltd

The Hierarchy command is used in order to calculate all of the sections of the structure and to perform Consolidations automatically, Figure 7.

It is worth pointing out that this technique is equally valid for a structure containing many hundreds of sections and not just the simple case considered in this demonstration.

Hierarchical Consolidation.

Thus through utilisation of the HIERARCHY feature of the system we are able to define a consolidation method that is easily defined and maintained since it does not rely on any programming.

In addition, by merely defining additional "hierarchies" we have the ability to consolidate the company data from different points of view, for example product type, geographical area etc.

Further, the ability to automatically carry out currency conversion, to perform full or partial consolidation and to carry out allocation of data throughout the structure are additional possibilities that require no programming.

Finally, the ability to produce comparitive reports from selected sections of the structure is enhanced. Thus a full report for each section, Figure 8, and a comparison by time period or by row, Figure 9, are immediately available.

```
System>STRUCTURE
Hierarchy File:/ABUDH2
Logic file:/ABUDL2
Data file:/ABUDD2
First,last row in section zero?1,3
First input row,last input row,last row to consolidate?12,80,80
Currency conversion required?NO
Inflation calculations required?NO
System>
```

Provide relevant information before performing the consolidation

```
System>EDIT /ABUDD2
*Section 1
*LIST

ROW      12
          13.00          12.00          11.00          12.00

ROW      14
        1000.00        1010.00        1020.00        1030.00

ROW      16
         150.00         154.50         159.14         163.91

ROW      22
          30.00          30.00          30.00          30.00

ROW      24
          45.00          45.00          45.00          45.00

ROW      26
          10.00          10.00          10.00          10.00

ROW      28
           5.00           5.00           5.00           5.00

ROW      46
       21666.67          20000       18333.33          20000

ROW      48
          16250          15000          13750          15000

ROW      50
       10833.33          10000        9166.67          10000

*END
System>
```

Section 1 contains the Manchester budget data.

```
System>HIERARCHY FULL
System>
```

The logic is calculated for each section in the hierarchy after aggregating relevant input rows.

FIGURE 7

```
System>REPORT USING /ABUDHR                           Recall summary report
                                                      specifications
System>DISPLAY SECTION
                                                      Display only required
                                                      sections. In this case, the
sections?1,2,7                                        Components Division (section
                                                      1, 2 and 7)


09:55:30 on 13/12/80

                              MANCHSTR

                   QTR1      QTR2      QTR3      QTR4      YEAR

SALES REVENUE     150000    156045    162318    168826    637189

CONTRIBUTION       60000     65145     70518     76126    271789

NET PROFIT         11250     20145     29268     31126     91789
                  ======    ======    ======    ======    ======
RTRN ON SALES %    7.50%    12.91%    18.03%    18.44%    14.41%

09:55:31 on 13/12/80

                              NEWCASTLE

                   QTR1      QTR2      QTR3      QTR4      YEAR

SALES REVENUE     600000    609000    618000    627000    2454000

CONTRIBUTION      424000    430360    436720    443080    1734160

NET PROFIT        338000    337060    335785    333122    1343967
                  ======    ======    ======    ======    =======
RTRN ON SALES %   56.33%    55.35%    54.33%    53.13%     54.77%

09:55:32 on 13/12/80

                              COMPNENTS

                   QTR1      QTR2      QTR3      QTR4       YEAR

SALES REVENUE     750000    765045    780318    795826     3091189

CONTRIBUTION      484000    495505    507238    519206     2005949

NEW PROFIT        349250    357205    365053    364248     1435756
                  ======    ======    ======    ======     =======
RTRN ON SALES %   46.57%    46.69%    46.78%    45.77%      46.45%
```

Computer Museum

FIGURE 8

```
SYSTEM>DISPLAY SECTION COLUMN 5 TITLE
Title:1981 BUDGET
sections?3-5,8,9
```

Comparative report for any particular column (column 5 - annual total in the example). Compares Engineering Division with K.F. Ltd.

```
09:57:29 on 13/12/80

                         1981 BUDGET

                   LEEDS     CROYDON   BRISTOL   ENGNRNG   K.F.LTD

SALES REVENUE     152250     1773000    497475   2422725   5513914

CONTRIBUTION       85260     1205640    294800   1585700   3591649

NET PROFIT        -17416      978692     91587   1052863   2488619
                 =======     =======    ======   =======   =======

RTRN ON SALES %  -11.44%      55.20%    18.41%    43.46%    45.13%

System>REPORT USING /ABUDR2

System>DISPLAY SECTION BYROW 54
```

Comparative report comparing any model row (row 54 - PROFIT in this case)

```
10:01:03 on 13/12/80

                          PROFIT REPORT

                   QTR1      QTR2      QTR3      QTR4       1981

1 MANCHSTR         11250     20145     29268     31126      91789
2 NEWCASTLE       338000    337060    335785    333122    1343967
                  ------    ------    ------    ------    -------

3 LEEDS            -4500     -4590     -6006     -2320     -17416
4 CROYDON         253500    247840    242302    235050     978692
5 BRISTOL          22500     24000     24185     20902      91587
                  ------    ------    ------    ------     ------

7 COMPNENTS      349250    357205    365053    364248    1435756
8 ENGNRNG        271500    267250    260481    253632    1052863
                 ------    ------    ------    ------    -------

9 K.F. LTD.      620750    624455    625534    617880    2488619
                 ======    ======    ======    ======    =======

System>
```

FIGURE 9

Automated Management Reporting.

If desired a system utilising Menus may be created within FCS-EPS such that appropriate models can be operated by unskilled staff.

One such situation might be the quarterly management reporting associated with the monitoring of the actual performance of the organisation.

In the case of K.F. Ltd, a prompting system has been defined that allows from a simple question and answer session a selection of reports to be produced, which incorporate data from the budget file, last years actuals file and this years actuals from the Nominal Ledger. The selected reports are by Factory and Period, and only valid combinations are available, see Figure 10.

Conclusion.

A demonstration of the FCS-EPS Decision Support System has been given, during which a simple quarterly budgeting model has been produced for the fictitious organisation K.F. Ltd.

The demonstration has illustrated the basic component steps in the creation of a decision support computer model and shown how such a model may be used to examine the uncertain future in a detailed manner that manual methods are unlikely to allow.

In addition, we have seen how a basic two dimensional system of financial variables and time periods may be extended into the third dimension of hierarchical consolidation.

Finally, it has been shown that by "black-boxing" a model of this type the benefits of the computer may be enjoyed by staff with the minimum of expertise.

```
WELCOME TO THE KF QUARTERLY UPDATE AND REPORTING SYSTEM.

OPTION ? HELP
OPTIONS ARE REP-REPORT QUIT
OPTION ?REP

PLEASE ENTER FACTORY NUMBER
   1 - MANCHESTER
   2 - NEWCASTLE
   3 - LEEDS
   4 - CROYDON
   5 - BRISTOL
ENTER NUMBER NOW = 1

QUARTER ENDING ? MAY
   INVALID -  CURRENT QTR IS SEPTEMBER
QUARTER ENDING ? SEP

DO YOU WANT SUMMARY REPORT (Y OR N) ? YES

KF QUARTERLY SUMMARY REPORT - QUARTER ENDING SEPTEMBER

MANCHESTER FACTORY
                                                             % VAR    % VAR
LAST YEAR    BUDGET                                ACTUAL    (BUD)     (LY)
---------    ------                                ------    -----    -----
 136290      156050     SALES REVENUE              155200      (1)       14

  27340       30300     VARIABLE LABOUR             31320       3        15
  40100       45450              MATERIAL           42860      (6)        7
   9270       10100              CARRIAGE           10450       3        13
   3605        5050              ENERGY              4836      (4)       34
   ----        ----              ------              ----      ---       --
  55975       65150     CONTRIBUTION                65734       1        17

  18900       20000     OVERHEADS PRODUCTION        22610      13        20
  15248       15000               SALES & ADMIN     13490     (10)     (12)
  11600       10000               R & D             11400      14       (2)
  -----       -----               - - -             -----      --       ---
  10227       20150     PROFIT                      18234     (10)       78
   7.50%      12.91%    PROFIT%REVENUE              11.75%

 REPORT FINISHED
OPTION ? QUIT

KF QUARTERLY UPDATE AND REPORTING SYSTEM TERMINATING
```

FIGURE 10

REMOTE LINE-PRINTING

by

Michael P. Mansfield

Managing Director, A1 Peripherals Limited

During the last five years there has been an increasing trend towards moving the line-printer out of the machine room and into end-user environments. This paper examines this trend and describes in some detail A1 Peripherals' unique offering in the remote printing development. By remote printing we mean high-speed, high volume printing accomplished by impact or non-impact line-printers at a distance from the CPU.

First it should be noted that remote printing is nothing new. Minicomputers have supported serial printers through their communications ports for many years. These devices fall into two categories, viz the classic KSR teleprinters such as the Teletype 33, and second the receive only printers such as the Digital Equipment LA 180.

As users' needs outgrew the capabilities of the serial printer both in through-put and in duty cycle requirements the obvious solution has been to replace the serial printers by line-printers. This has commonly been done in three ways. First, serial asynchronous interfaces such as RS232, i.e. speeding up the existing link. Second, long lines for paralled connections up to 1000 ft. Third, remote batch protocol emulation, such as IBM 2780 or 3780. We now introduce a fourth solution which is the Remote Line Printer System (RLPS). This combines many advantages of the existing three methods while adding several of its own.

When choosing a remote line-printing solution there are five major considerations that should be bourne in mind. The distance of the printer from the CPU; the duty cycle needed; the through-put required; the price sensitivity of the application; and the level of data integrity required. In a few applications these five criteria may be equally important. However two factors, the distance from the host computer and the data integrity required, usually outweigh the others.

Consider, then, the advantages and disadvantages of each solution. First, long lines. The primary advantage of the long lines solution is that it imposes no operational change or other overhead upon the system. The printer can be driven at maximum through-put just as if it were a local printer. There is no additional cost for communications hardware or line charges. The disadvantages are that long lines are limited to around 1000 ft; the labour and problems of laying such a long cable; the cable expense itself (approximately £1 a foot); and the inflexibility of moving the printer once installed.

Second, the asynchronous serial interface. RS232 is the most commonly used serial interface and has become the de facto standard for mini and micro computer remote connection. However, for line-printer applications this method has serious disadvantages. A heavy asynchronous load can degrade a minicomputer's performance by as much as 30%. This occurs because, in most asynchronous serial environments, the CPU is interrupted for each character transfer, while for line-printers the interrupt is only once a line. This means also that the maximum speed that can be obtained from, say, a 1000 line/minute printer over a 9600 baud line is only 550 lines/minute. Data integrity in this environment also suffers, unless costly intelligent modems or multiplexors are used to effect error detection and re-transmission. For a given speed of printer,serial asynchronous interfaces also require costlier, higher speed modems than does our own RLPS system.

To make full use of modern high-speed line-printers users often want to take advantage of features such as direct access vertical format units (DAVFU), paper slewing, variable line pitch, etc. These characteristics are not available on serial devices. This may necessitate special communications hardware and software such as device handlers which tend to be a very specialised thing to produce. In addition separate versions of the applications software may be necessary for both local and remote printing.

Thirdly, the synchronous protocols. High-speed remote batch printing has evolved primarily in the main frame environment. In this configuration remote printers are interfaced to the host CPU over telephone lines. Each mainframe manufacturer supports remote processing through its unique protocols.

In the minicomputer environment some users have elected to emulate mainframe protocols to benefit from their error detection and re-transmission features and higher through-put than asynchronous interfaces.

Users who go for this solution are limited to one protocol per remote line printer. As with serial asynchronous interfaces this approach requires special communications hardware and software, and usually also means separate versions of the applications software for local and remote printing.

The user should be well educated in the protocol he elects to use since he will usually have to develop his own programs. While synchronous protocols offer a superior alternative to RS232, they usually place more overhead on the CPU and certainly more than our own RLPS. They often require additional and expensive hardware, such as synchronous interface boards.

So now let us examine a new alternative to the systems so far discussed, the Remote line Printer System (RLPS). This system has been developed over the last 12 months by Digital Associates Corporation of Connecticut and is supplied and supported here by A1 Peripherals Limited.

The RLPS appeals to banks, service bureaux, and multi-location companies where a volume of high-speed printing is required in remote locations.

Figure 1 shows a typical configuration. Operating through the parallel printer port of the CPU, the RLPS configuration consists of a parallel-to-serial transmitter, a serial-to-parallel receiver, a printer, printer controller, and modems or multiplexors as required.

Figure 2 is a block diagram of the RLPS itself. The microprocessor-based RLPS transmitter unit accepts data in a paralled mode, converts it to serial data, blocks and compresses it, checks for errors, and then transfers the serial data to the modem.

At the remote location, the nearly identical RLPS receiver accepts the serial data, converts back to parallel, unblocks and decompresses it, again checks for errors, and transmits it to the line-printer.

The RLPS can be connected in a number of different ways. Direct connect configurations (see Figure 3) are for applications within the same building or site. The RLMS units are connected by a cable providing two twisted pairs. Total cable length is about 4-5000 ft. In the direct connect configuration the RLPS may be operated up to 56K baud for maximum through-put.

In a private leased line configuration (see Figure 4) the RLPS operates via a leased 4 wire telephone circuit and RS232C synchronous modems or multiplexors.

In some cases it is feasible to use local area data sets or line drivers (often up to 6 miles) as shown in Figure 5.

Finally the switched network allows use of the normal telephone system (see Figure 6). Auto dialling equipment may also be used and this system is ideal where print volume does not justify a leased line or where jobs are printed from one or more CPU's to one or more remote printers.

So what are the advantages of the RLPS?

The single biggest advantage of the RLPS is a significant reduction in communications costs associated with remote printing. By accepting data from the CPU in the more efficient parallel mode, compressing it, and transmitting in a block mode fashion, the RLPS can double or triple the through-put of RS-232 connected line printers. This improved through-put results in lower line costs in a dial-up environment and in a dedicated line application, it frees up the line for other tasks. Additionally, the RLPS permits the user to reduce communications baud rate and modem requirements:

2,400 Baud = 300 lines per minute.

4,800 Baud = 600 lines per minute.

9,600 Baud = 1,000 lines per minute.

For applications requiring even greater through-put, the RLPS can be operated at 19.2K baud for speeds up to 1800 LPM.

Unlike a serial asynchronous environment, a dial-up line can be used up to 4,800 baud, again resulting in savings.

Because the RLPS is the only existing product to operate from the printer port, no special communications hardware or software is required. It looks like a local printer to the CPU.

All users have the same version of the applications program, and all line printer functions remain available. No reprogramming is required at the host or remote sites for the addition of multiple remote users, since the same applications software is used in both.

By assuming the burden of the communications protocol/overhead, the RLPS virtually eliminates the host processor degradation which occurs when a high-speed line printer is remotely connected through the minicomputer's comm port.

The RLPS employs automatic line hit/error detection and re-transmission. This feature not only ensures data integrity for critical applications, but permits the use of less costly "dumb" modems.

In summary the comparison of RS232 versus RLPS is shown in Figures 7 to 9.

As an example, the RLPS is being used by a large US East coast service bureau in the banking environment. Because of the high volume of printed output necessary they could not drive a high speed remote printer at the required speed at subscriber locations using the RS232 interface. Their solution to this problem was to do all their volume printing on second shift, then hand carry printout to the airport during the third shift and fly the output to subscriber sites. Certainly an expensive solution.

Their second solution was to place a separate minicomputer at each of their subscriber locations in order to communicate with the central site CPU and drive a high speed line-printer. Now with the RLPS they have high-speed, high-quality output at remote subscriber locations without the expense of a minicomputer - in this case a $30,000 saving per site.

The RLPS has a couple of useful extra features. There is a programmable auto dialling system which stores up to 50 'phone numbers. There is an auxiliary asynchronous channel which provides a stat. mux. like circuit, permitting a terminal to be installed at the printer site, creating a "mini" RJE station.

Future additions to the product will include more auxiliary channels and a remote spooling system. This will use a Winchester disk to store print data when the printer is off-line, or when the data flow exceeds the printer speed, or to hold and reprint data files that have already been transmitted.

The only disadvantage of the Remote Line Printer System is its initial cost of £4,500. However, we believe this will be quickly recovered as the RLPS represents the only cost effective solution for high volume remote printing applications where data integrity and minimising communications costs are paramount.

Michael P. Mansfield

Michael Mansfield is managing director of A1 Peripherals Limited, independant suppliers of the largest range of line-printers and interfaces in the U.K. He has worked extensively in the computer and peripherals business including being sales manager for Dataproducts, UK and district sales manager for Hewlett-Packard.

A TYPICAL RLPS SYSTEMS CONFIGURATION

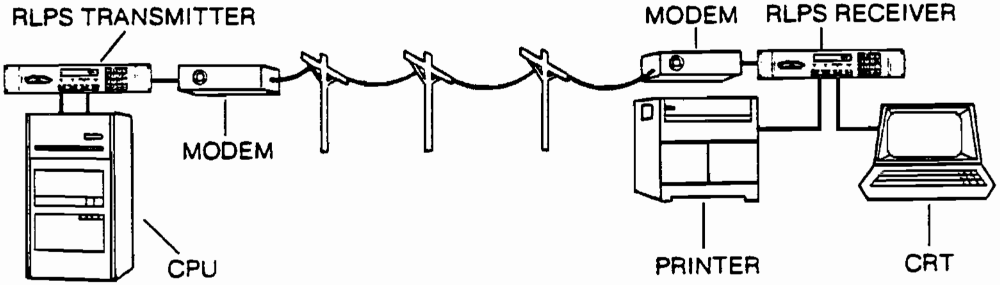
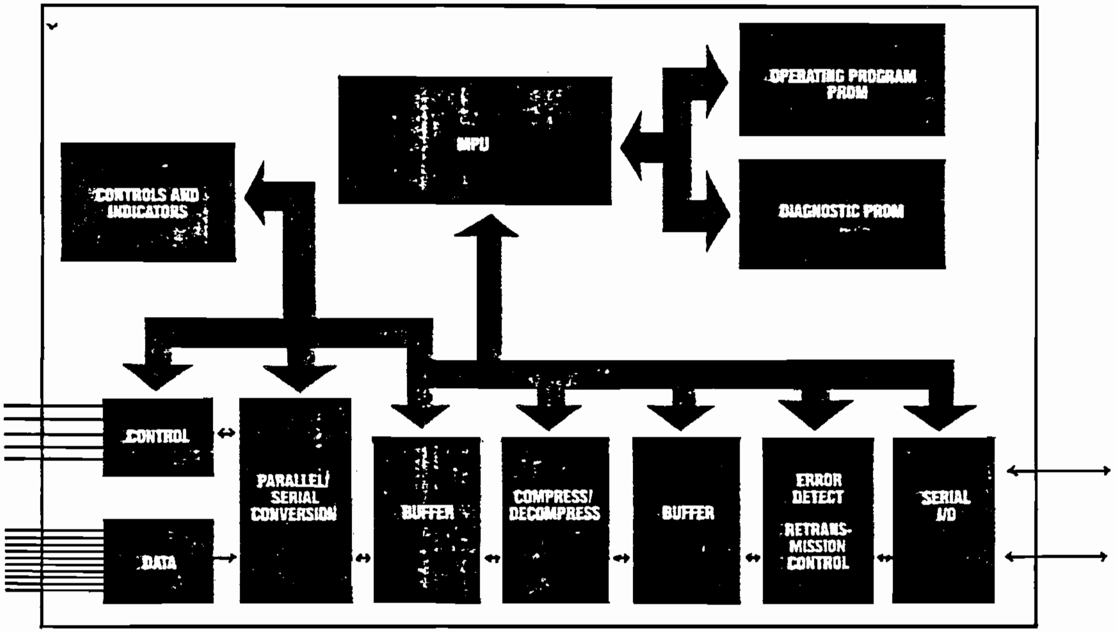

RLPS TRANSMITTER
MODEM
CPU
MODEM
RLPS RECEIVER
PRINTER
CRT

FIGURE 1

HOW IT WORKS:


OPERATING PROGRAM PROM
MPU
DIAGNOSTIC PROM
CONTROLS AND INDICATORS
CONTROL
DATA
PARALLEL/ SERIAL CONVERSION
BUFFER
COMPRESS/ DECOMPRESS
BUFFER
ERROR DETECT
RETRANS-MISSION CONTROL
SERIAL I/O

RLPS TRANSMITTER/RECEIVER

FIGURE 2

RLPS DIRECT CONNECT

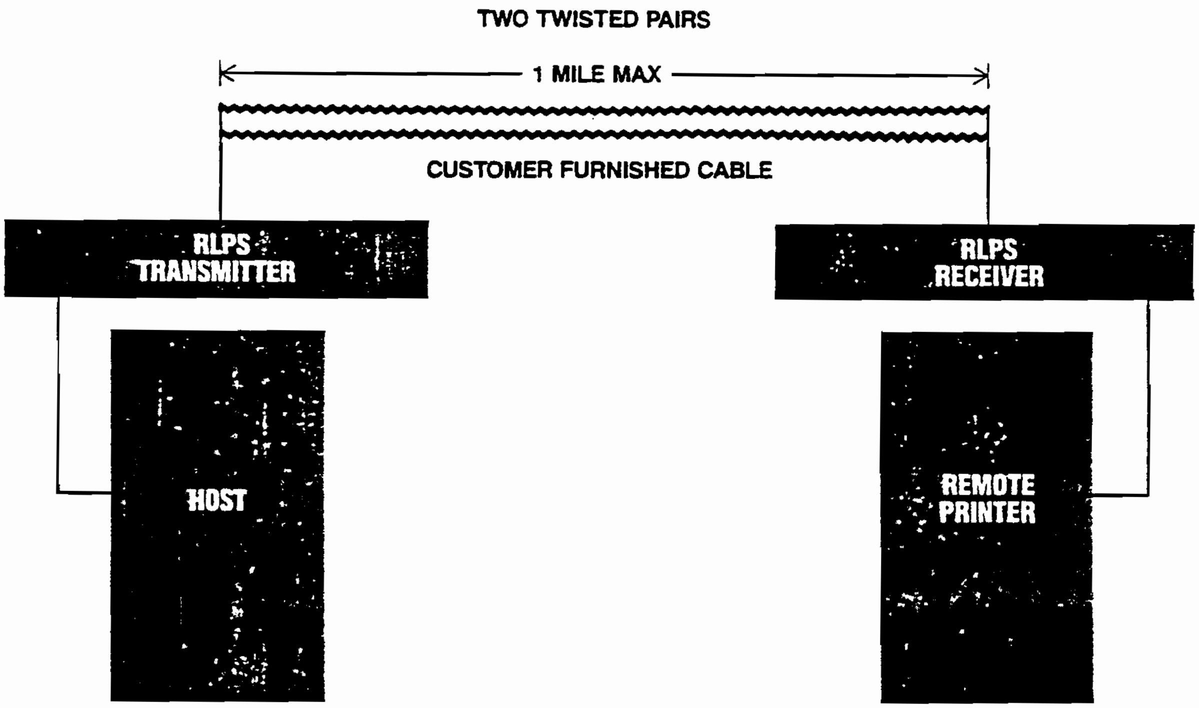

TWO TWISTED PAIRS
1 MILE MAX
CUSTOMER FURNISHED CABLE
RLPS TRANSMITTER
RLPS RECEIVER
HOST
REMOTE PRINTER

FIGURE 3

RLPS LEASED LINE CONFIGURATION

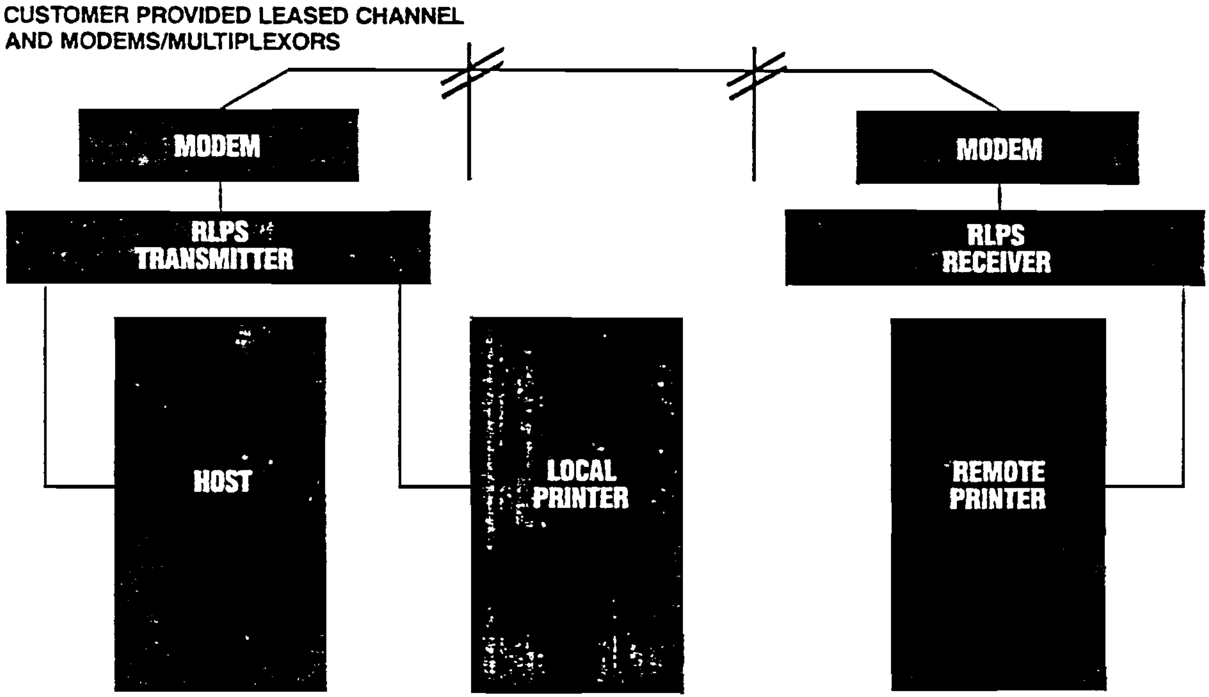

CUSTOMER PROVIDED LEASED CHANNEL
AND MODEMS/MULTIPLEXORS
MODEM
RLPS TRANSMITTER
HOST
LOCAL PRINTER
MODEM
RLPS RECEIVER
REMOTE PRINTER

FIGURE 4

RLPS LIMITED DISTANCE (6 MILES) INTERNAL MODEM CONFIGURATION

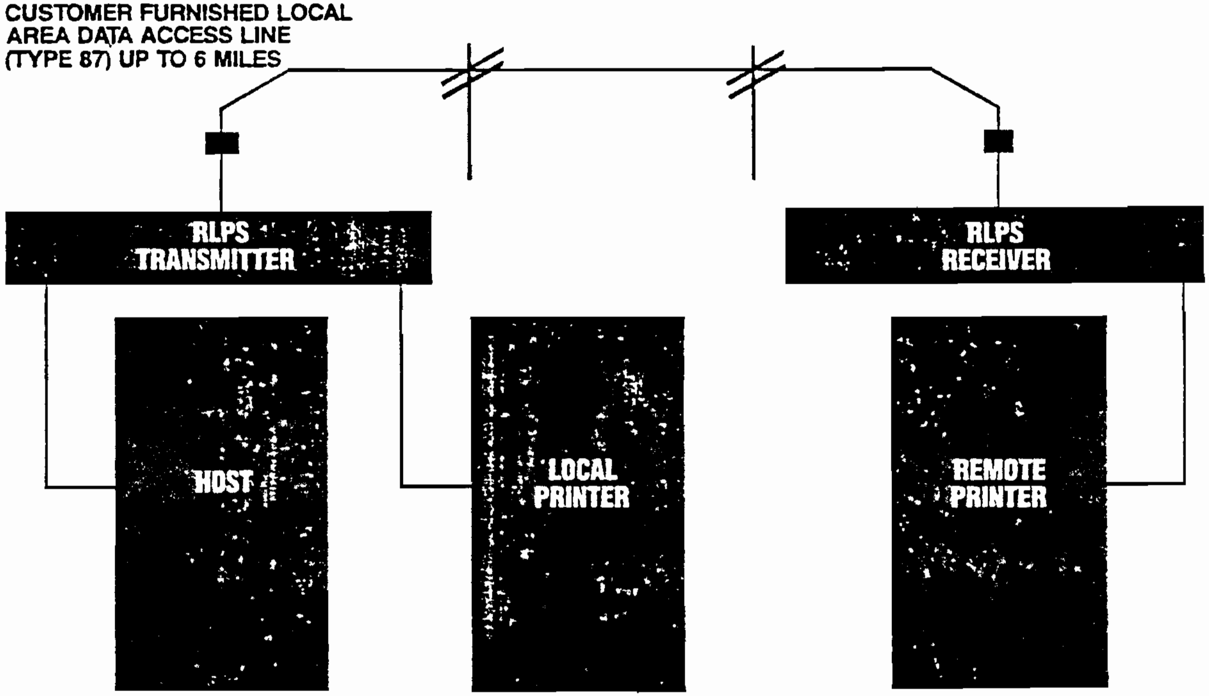

CUSTOMER FURNISHED LOCAL
AREA DATA ACCESS LINE
(TYPE 87) UP TO 6 MILES
RLPS
TRANSMITTER
RLPS
RECEIVER
HOST
LOCAL
PRINTER
REMOTE
PRINTER

FIGURE 5

RLPS DIAL UP REMOTE CONFIGURATION

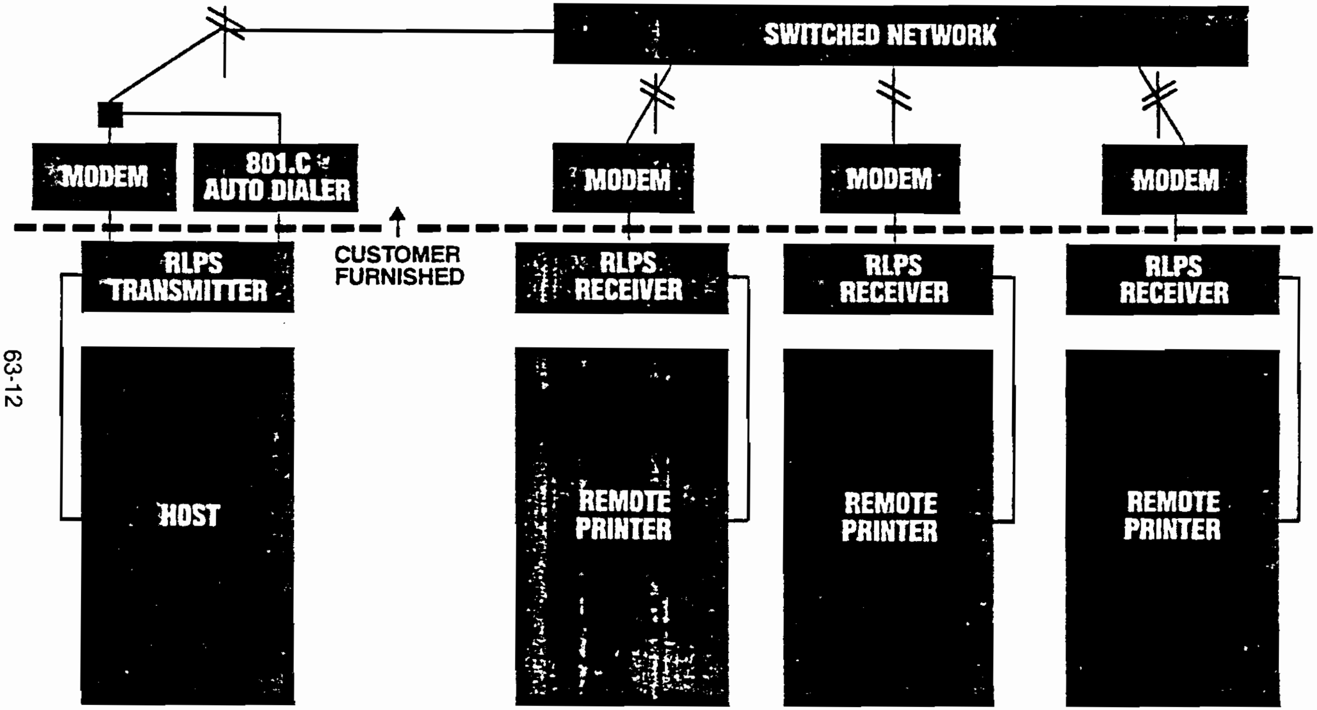

SWITCHED NETWORK
MODEM
801.C
AUTO DIALER
MODEM
MODEM
MODEM
CUSTOMER
FURNISHED
RLPS
TRANSMITTER
RLPS
RECEIVER
RLPS
RECEIVER
RLPS
RECEIVER
HOST
REMOTE
PRINTER
REMOTE
PRINTER
REMOTE
PRINTER

FIGURE 6

RS-232 V.S. RLPS

• At 2400 baud you get approximately 190 LPM	• At 2400 baud you get 300 LPM.
• At 4800 baud you get approximately 375 LPM	• At 4800 baud you get 600 LPM.
• At 9600 baud you get approximately 550 LPM	• At 9600 baud you get 1000 LPM. .

FIGURE 7

RS-232 V.S. RLPS

• No error detection of line hits or transmission faults.	• Automatic error and line hit detection, as well as automatic retransmission.
• Requires special communications software handlers.	• Standard operating system software, no specials.
• CPU degradation up to 30% for 1000 LPM line printer.	• No CPU degradation at any speed.

FIGURE 8

RS-232 V.S. RLPS

• Requires higher speed modems and costlier lines.	• Saves money and time on both modems and lines. Cuts costs in half!
• Frequently requires separate versions of applications software for local and remote users.	• All users have the same version of the applications program.
• Often requires special communications hardware for the CPU.	• No special communications hardware required for the CPU.

FIGURE 9

HEWLETT PACKARD
Systems Performance Team, SYSTEMS MARKETING CENTER
19447 Pruneridge Avenue, Cupertino, CA 95014, USA 408-725-8111

Technical Publication Costs Cut In Half with Laser Printing

Steve Wilk & Sam Boles
Hewlett-Packard

Synopsis

Technical publications enjoy all the standard maladies of other publications: the high cost of type-setting, the rising cost of printing, the long turn-around time from camera-ready copy to multiple copies ready for distribution. But technical publications tend to be unique -- at least in degree -- in their volatility and the urgency of timeliness.

Hewlett-Packard, by being in the computer business, is by definition in the technical publication business. An HP team of development engineers and performance engineers have implemented a methodology using HP3000 document processing facilities and the HP2680 laser printer to cut time costs and dollar costs in half for internal-grade technical publications.

The Problem

Where It All Begins

This story begins in the Performance Center of the Hewlett-Packard computer facility in Cupertino, California. That's where we work as part of the Systems Performance Team running customer benchmarks and doing the performance characterization of many of Hewlett-Packard's hardware and software products.

Big-Ticket Benchmarks . . .

As any of you who've done a benchmark know -- perhaps too well -- they're *expensive*. A complex benchmark emulating a hundred-terminal interactive load across a range of job mixes and configurations can cost *tens of thousands* of dollars -- even *hundreds of thousands*.

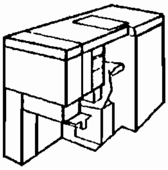
. . . and What They Buy

Thru such benchmarks our Systems Performance Team generates some very valuable intelligence on a variety of HP 3000 configurations as well as performance optimization techniques.

This intelligence has a flavor to it that is uniquely *real-world*. The benchmarks are run with *customer data* against *customer files* that often are taken directly off a *customer's production* system.

This performance intelligence is also *leading edge*. The benchmarks are often run on newly-released or pre-release products. The environment is such that we can experiment with configurations to calibrate

using the HP2680 Laser Printer and Hewlett-Packard Solution Software on the HP3000 Computer



resource sensitivity: changing, running, measuring time after time, validating or rejecting our hypotheses, each time understanding the subtle interplay of system concurrencies a little better - or at least appreciating the *limits of our understanding* a little better.

The environment also enables us to deploy *specialists* on special issues. If it's an ATP timing delay issue or a VIEW screen download issue, we can get the lab expert - perhaps in the next building - to bring to the problem the level of expertise unique to the *author* of the code or the *designer* of the board.

A Central Clearing-House

The Performance Center then is the forum for the early encounters of our products and your world. It has the hardware and software resources, the human resources and the instrumentation and metrics to derive valuable intelligence from these encounters.

So What's The Problem?

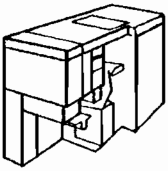
The problem is *disseminating* that valuable intelligence. *Sharing* with the HP support organization around the world, and, thru them, the customer base around the world. *Leveraging* the heavy investment in performance characterization of HP hardware and software products. Basically, *spreading the word*.

The Birth of pn^2

Last year we decided to establish a formal mechanism to communicate our performance information to the field. It's name ended up as *Performance News Notes* -- or pn^2, as we affectionately nick-named it. It's a monthly publication, 10 to 20 pages in length (for 1-shot reading). It's current circulation is 2600 readers in the HP organization around the world.

In our original specifications we set out some objectives for the publication:

> *Quality*: It didn't need to be slick and in color, but it did need to be readable and professional in appearance.
>
> *Cost*: We always aim at zero and compromise to the extent that quality and other necessities dictate.
>
> *Speed*: It had to be fast. Especially with new product characterization we couldn't tolerate a multiple-day multiple-iteration typesetting cycle, and a 4-day to 2-week printing cycle. But our cost objectives wouldn't allow the premium to expedite these cycles. Also we wanted to use the HP bulk mail system to minimize national and international distribution costs, so labeling and sorting speed were another consideration.
>
> *Flexibility*: It had to have graphics capability since much of our information is best communicated in graph form. It needed multiple character sets for more intelligible typography.



using the HP2680 Laser Printer
and Hewlett-Packard Solution
Software on the HP3000 Computer

The Solution -- Set-up Phase

The Laser Printer

We decided to try the *Laser Printer*. It looked like a reasonably good fit for our quality, speed, cost and flexibility objectives. We had an HP2680A on one of the systems in our department, so we gave it a go.

Getting Started with IDS/3000

We spent a few hours getting set up. We used IDSCHAR ("IDS" means Interactive Design System) software and a 2647 graphics terminal to design our logo. That took a couple of hours, counting the tweaking that you're always tempted to do even though *you're the only one* who'll ever notice that one dot out of the 180 per inch that's a little bit out of line.

Then we moved into forms design with IDSFORM software and the 2647. We needed a title page and a mailing label page. This took about 2 hours, plus another hour or 2 of moving things around till everybody -- or almost everybody -- was reasonably happy.

IFS/3000 for Device Specifics

IDSCHAR and IDSFORM generate dot matrices and vectors that are device-independent. This data is then compiled by IFS2680 ("IFS" means Interactive Formatting System), into an ENVIRONMENT file that is compatible with a specific I/O device, eg the HP2680 laser printer. So we used IFS2680 to compile our ENVIRONMENT specifications into an ENVIRONMENT file that the HP3000 could then download to the laser printer.

The ENVIRONMENT file loads character sets (up to 32 different character sets at a time) and forms (up to 32 different forms at a time) into the memory (we have a megabyte of main on our HP2680) that's used by the processor in the laser printer to control the laser so you get an 8-point Helvetica Bold instead of a 10-point Roman Italic. Building the ENVIRONMENT file took another hour or two till we were satisfied.

In all we probably spent about a day getting set up. Some of this time was *learning curve* and some was that "one last touch" of the *amateur artist*, but most was the time necessary to design the form or logo and enter the points and other specs.

The real advantage was the fact that we could do it all ourselves without the start-stop iterations, queues and communications entropy of a graphics department. The initial set up was done in *one continuous uninterrupted block* of time. There were no meetings or key resources to schedule. There was no "now let me see where I was two days ago when I last worked on this" re-think time.

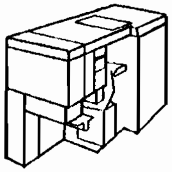
(Of course, the only time you can get that big a block of uninterrupted time is on weekends, so the one-day set up time didn't cost the company anything.)

Typesetting and Composition

We chose TDP (Text and Document Processor) for this because we wanted 2-column capabilities (to enable speed readers to bounce only vertically) and right justification of proportional character sets (to get more balanced "Linotyping"). Also, we needed the capability of including charts from DSG (Decision • Support Graphics) integrated with the text.

We included several character sets in our ENVIRONMENT file:

using the HP2680 Laser Printer
and Hewlett-Packard Solution
Software on the HP3000 Computer



Helvetica Bold: 8-, 14-, 24-point
Helvetica Italic: 8-point
Roman Bold: 10-point
Roman Italic: 10-point
Line Printer: 8-point
Math: 12-point
Inverse Line Printer: 12-point

We specified the appropriate FONTID's to TDP to couple with the ENVIRONMENT file, specified margins, columns, page length, and the like and proceeded to do our typesetting and layout.

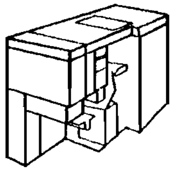
For the layout of charts (bar/line/pie) from DSG and illustrations from HPDRAW, we found that we were doing a lot of trial and error placement, so we set up a template like this:

```
/* ILLUSTRATE rasterfile
/NAME RASTER rasterfile
/ILLUSTRATE figurefile:figure #lines
```

We would NAME the raster file each time we changed the size or the figure (having explicitly purged the old raster file as needed) and therefore had to do another vector-to-raster conversion via the ILLUSTRATE command. Then for the next pass we'd comment out the NAME and second ILLUSTRATE, and de-comment the first ILLUSTRATE so we pulled in the raster form without having to do a vector-to-raster conversion when not needed:

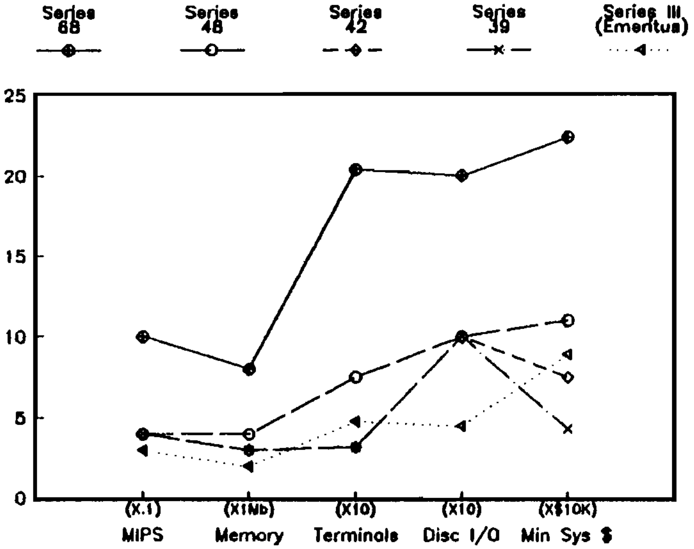
```
/ILLUSTRATE rasterfile
/* NAME RASTER rasterfile
/* ILLUSTRATE figurefile:figure
     #lines
```

This facilitates getting charts like the following integrated with the text:



using the HP2680 Laser Printer
and Hewlett-Packard Solution
Software on the HP3000 Computer

HP3000 Performance Profiles

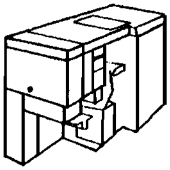


(graphics by DSG/3000)

This template technique we extended to the issue level, so for the subsequent issues the engineer who had responsibility for that issue could "cookbook" from the archive file of the previous issue without re-inventing the TDP commands. This saves time and gives consistency to the appearance of the publication.

We ran through a number of iterations on our first issue, balancing artwork, policing "widows" (TDP can do a lot of this automatically), cleaning up typo's and again

using the HP2680 Laser Printer and Hewlett-Packard Solution Software on the HP3000 Computer



falling into the "one last touch" syndrome of the amateur artist. Once again we found a significant *productivity surge in cutting out the middleman* -- this time the typesetter. We could do a lot of the local typesetting on the fly as we wrote the articles at the terminal. The global typesetting and layout commands were largely templated (about 75%) from the previous issue. The cut-and-paste we did by computer. And the inevitable last-minute "stop-the-presses" newsflash could be accommodated with only a little pain in a matter of minutes rather than days via the typesetter's queue.

Then finally, there we were with our first issue, camera-ready. We had done the logo and forms design, the typesetting, the graphics and the layout. We were ready for the printer.

We went to our in-house offset service. They were booked solid for the next week and a half. That was 10 days. We couldn't wait that long. If we paid the expedite premium we could get it by the end of next week. That was 5 work days. Better, but still *not good enough*.

What to do?

It was then that someone said, "Why not do it on the laser?"

"The whole thing?" we said.

"Why not? It prints 45 pages a minute."

The next day was Saturday. There's only sporadic use of the laser printer weekends (especially since we had finished our prototype the previous weekend). We needed 1300 copies at 12 pages each. 12 * 1300 = 15,600 pages. 15,600/45 -> 347 mins -> 6 hours.

It looked do-able . . . were we in for an unpleasant surprise.

The Solution -- Manufacturing Phase

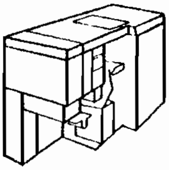
We decided to give it a go. That Saturday morning, we set OUTFENCE 10 to prevent interleaving by random print-outs and to enable us to drain the queue as needed but in a controlled fashion. We did a HEADOFF 14 to save paper and reduce the splitting work. We did a TDP FINALQ of our text file to suppress the TDP message at the end. We set COPIES=120 on the SPOOL file to get multiple copies. And we were off and running.

But not for long.

We found that after the first copy, the whole SPOOL file (including the heavy-duty ENVIRONMENT portion) was being down-loaded. Because our ENVIRONMENT file had so much in it, the laser printer was going into a warm-up cycle between each copy. (To conserve energy, the infra-red fusing drum cuts off if not in use for a few seconds, then it has to warm up to the fusing temperature before the printing resumes.)

This cut our print rate to 10 pages a minute. Our 6 hour print job now looked like 25 -- if everything went well. And Mr. Murphy, the silent partner in all such pioneering ventures,

using the HP2680 Laser Printer
and Hewlett-Packard Solution
Software on the HP3000 Computer



said that was not likely to happen. Besides, we were getting an extra page from TDP and that meant every other copy had to be refolded to get the cover on the front. (We used an even number of pages to prevent this but it wasn't working.)

SPOOK to the Rescue

We were just about to hang it up and wait the 10 days for the "real" printer to do the job when someone said, "Why not use the SPOOK APPEND command, build a new SPOOL file with one copy of the ENVIRONMENT and multiple copies of the text. We can even cut out the extra page in the process."

We tried it and it worked. The print speed was up to 35-40 pages a minute -- not far from the 45 ppm rating of the HP2680. We decided to give it another try.

But before we got started, someone said, "How about those name-address labels we were going to print separately on the gummed stock -- why don't we just splice them in while we do the SPOOK APPEND to solve the performance problem?"

Labels on the Fly

We had already tried the TDP MAILER facility to do this. The MAILER is good for multiple addressees for text that does not have a heavy FINAL formatting load, since its formatting performance is about O(n). With 12 pages of 2 column right-justified proportional text plus 8 graphics inserts, we were taking several minutes for a FINAL even with the graphics already in raster format. An O(n) performance would give us 1300 FINAL's at several minutes each. This was not operationally feasible, but a SPOOK APPEND splice might be.

We tried it. The redundant APPEND's (you're APPENDing a text set for each addressee and an ENVIRONMENT set for each 10-15 addressees) builds at a slower (about 2x) rate than your printing rate. If you have enough temporary disc space, you can fire up the APPEND's Friday night, start your printing Saturday morning and have a fairly level work pressure for the operator who's doing the splitting, QA and boxing for the distribution center. And this does save the time for the label stick-on step.

It wasn't elegant, but the prototype was functionally and performance-wise feasible, so we launched it.

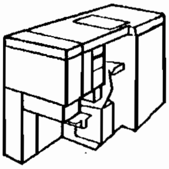
Short on Finesse, but it Worked

Well, with that prototype methodology, we managed to brute-force the first issue in about 18 hours. (The 18 hours includes some cockpit errors I'd rather not talk about. Let's just say that engineers are not necessarily the best operators.) Not exactly the 6 hours we had dreamt of -- but we had beat our alternatives by a week or more and had established that we had the technology and it was just a Small Matter Of Methodology (remember "SMOP" of Large Systems Games fame: Small Matter Of Programming . . . well, this is "SMOM") that needed to be developed.

By the third issue (if you want to find out how short a month is, commit to getting out a monthly publication), we got a bit of prototype software and STREAM commands in place to make our brute force methodology a little more tolerable.

Later, we contracted a 17-year-old student who's working his way thru college doing laser typesetting and printing. Off-loading the grunt work to the student has freed up engineering resources that can be used to get the methodology to an operationally sound condition.

using the HP2680 Laser Printer
and Hewlett-Packard Solution
Software on the HP3000 Computer



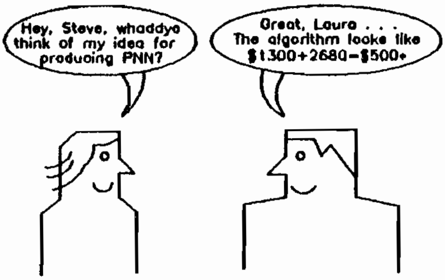
The Results

Structured Development

The approach we've used in evaluating laser technology for typesetting, composition and low/medium volume printing is an extension of Structurism. We're all aware of Structured Programming and Structured Design. This is Structured Development or Prototyping.

We did the first issue with engineers doing the operations, throwing together ad hoc software as necessary to establish that the technology is capable of producing the results. That's established now.

In the next step of development we prototyped some software in BASIC and PASCAL to automate some of the more onerous clerical procedures. The idea here was that if a viable methodology for production were not attainable, we would throw away less code of less value.


Hey, Steve, whaddya think of my idea for producing PNN?
Great, Laura . . . The algorithm looks like $1300+2680-$500+

(graphics by HPDRAW)

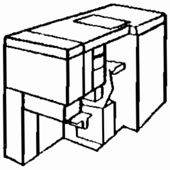
With the crude prototype in place for more than 6 issues, we've seen some interesting results:

12 page performance news notes
2 column right justified proportional
7 fonts, 8 pieces of artwork
(bar and line charts, illustrations)

7:00 PM Friday: last text edit
6:00 AM Monday: 1700 copies printed, pre-sort labeled, split
Variable cost: $600

Variable cost includes labor, paper, toner, carrier, drum cost and maintenance. The machine is idle otherwise, so the sunk costs

using the HP2680 Laser Printer
and Hewlett-Packard Solution
Software on the HP3000 Computer



of printer depreciation, real estate etc. are excluded.

As word spread on the results of laser typesetting, layout and printing, we applied the methodology to other publications. Here's a recent scenario that may have set a (world-class?) record:

Friday at 3:00 PM, a marketing manager asks if it's possible to have 1000 copies of a 20 page (5 character fonts, 2 column, 1 piece of artwork, no labels) strategy document by Monday (3 calendar days away).

Friday night: Logo design done, forms design started, preliminary environment file compiled.

Saturday: Sample of cover page and first text page built, reviewed approved. Text data not yet available as ASCII file.

Sunday 11:30 AM: Text data ready for typesetting to start.

Sunday 4:00 PM: Last edits made to text file.

Monday 3:00 AM: Last of 1000 copies printed using 3 laser printers (extras idle on graveyard shift Sunday night).

We normally call 5:00 PM Friday till 8:00 AM Monday 0 business days. Technically, 11:00 AM Sunday till 3:00 AM Monday is perhaps a turn-around of -1.5 days. In our experience, negative turn-around time is not all that common.

Futures

Our evaluation of the prototype is positive. We've saved at least $5,000 on *pn* alone, which annualizes at over $10,000.

We are currently evaluating a phase I production version that is operationally sound and simple and may have a 30-50% performance improvement (over the current label splicing methodology.) This would be non-supported contributed library class software.

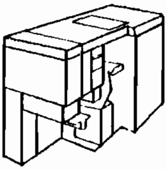
This same facility may be expandable to include a serial number of each page for sensitive "Do not copy" documents. The control file for this may be the name-address interface file enhanced. This would be non-supported contributed library class software.

Long-term, we see satellite communication to field offices of the "spool file" equivalent and the generation of hard copy as needed at the field location.

Conclusion

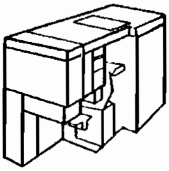
The fact that we are continuing to use the prototype version of the laser typesetting/printing methodology is evidence of its economic and functional feasibility. The fact that we are intending to invest the engineering to productionalize the methodology reflects that it has potential for material contribution in the near term.

using the HP2680 Laser Printer
and Hewlett-Packard Solution
Software on the HP3000 Computer



seblug12 1030 29JUL83

using the HP2680 Laser Printer
and Hewlett-Packard Solution
Software on the HP3000 Computer



HP3000 IUG Meeting - Edinburgh **PRESENTATION ABSTRACT**

New product announcements from Direct

Bernard Lovell

Overview

The personal computer has already moved into the corporate environment to address the needs of executives and managers for desktop computing. But the personal computer does not integrate well into the existing corporate computing environment. Integration has four components:

1. Running host application programmes as a VDU.
2. Data exchange between host and PC
3. Compatible data structures on host and PC
4. Integrated host/PC applications.

Integrated Personal Business Computing (IPBC) is the fulfillment of these four levels. Without IPBC, the performance of a personal computer may not justify the expense of intalling it on critical departments or offices.

Direct's solution is the 1025 and the new 1625 workstations. By investing in one of these IPBCs, the users gains both terminal and personal computing capabilities in one compact desktop unit. Extensive options include graphics, modem and hard disk storage for the 16 bit 1625. Compatible data base structure between the IPBC and the host 3000 allows information to be shared throughout the organization, without re-keying or duplication of data..

1625 specification

Terminal processing: full Direct 825 personality with block mode and 32K of video RAM. Terminal mode is selected by a single key combination, even within application programmes.

Local processing: Choice of MS/DOS or CPM/86 operating system running on an Intel 8088 microprocessor. CP/M80 8 bit code also executes via a Z80 CPU.

Local storage: twin floppy disks of 2 X 320 Kb, or Winchester with single disk backup (10,20 and 40MB configuration). Disk format is IBM PC compatible. Direct 1025 format is also supported by a soft-sense mechanism which reads and executes 8 bit programmes.

IBM PC compatibility: the 1625 allows the user immediate access to IBM PC software without conversion via IBM disk format, IBM compatible operating system, IBM/HP dual function keyboard and IBM screen addressing.

IMPROVING HARDCOPY OUTPUT FOR BETTER EFFICIENCY

By Matt Cuson
Hewlett-Packard
European System Printer Product Manager

Does your desk seem to attract paper at an ever increasing rate? Do people ask more from you, your job and your department than they did in the past? Do you feel constrained by the system regarding how you do your work? Remember when we programmed the systems to work for us; have they programmed us to work for them?

Sound familiar? Welcome to the exploding world of information. Computers generate an ever increasing amout of data. This increase leads to increases in information exchange AND increased pressure for individuals and companies to keep up. The pace will not slacken, it will accelerate. How can we cope? We cannot work beyond our physical limits; we must adapt to, the changing environment and take advantage of the resources we have available. By working smarter rather than just faster we can avoid becoming a casualty of the information explosion. This paper discusses ways to cope with and even control the information explosion by improving hardcopy output.

When the big push for computer systems first started, computers freed time for us to work on other aspects of the job. Automation yielded time savings by speeding up manual systems. Fewer errors were made, processing was done faster. Computer systems input, check, process, sort, store and, print data faster than we could ever hope to process manually. The proliferation of computer systems from micros to the super big and fast mainframes augments the already rapid pace of information processing and exchange. The result is that each day we face more information from more sources on more varied topics than the day before but our time per work day is still 8 hours (ok 9,12,14 hours).

Computers have helped us work faster. But alas, there is a limit. New systems implemented with the old approach will not yield the same significant results of the past. Computers, word processors, time management, Theory X, Y, Z, personal workstations for every employee, graphics and fast CPUs alone cannot contain the information explosion. Talk about productivity won't help. We need more. We need creative systems that take advantage of the technology AND allow the people to work they way they want to work.

The process which will help us work smarter is an extension of the the evolutionary process which brought us from manual to automatic systems. The goal is to work in such a way which effectively uses the increasing amount of information available. One way to achieve this ambitious goal is to select an area where benefits would be widespread. A good place to start on your venture to work smarter would be to examine hardcopy output usage in your organization.

Why hardcopy? Several reason makes hardcopy a good choice: 1) Everyone uses paper, 2) today's technology permits us to implement better systems NOW, 3) Results are quantifiable.

HARDCOPY AFFECTS EVERYONE

When evaluating ways in which to work smarter, people's knowledge of how they work with output will help identify keys areas subject to improvement. We may take the knowledge for granted but with time, creative new approaches to the work process can be uncovered.

I don't think anybody would argue with the fact that the vast majority of people use paper in their jobs. Not everyone uses computer generated output directly, but most would deal with memos, typed reports and manuals at some time. Because paper is so widespread (probably the most common element across jobs) a small savings among individuals can quickly grow into very large savings for the whole company. In addition, benefits would quickly be felt and compounded over the large user base.

Not only does output get used by nearly everyone but there are as many ways to use output as there are people to use it. This fact plays a very important role when it comes to implementing a system. Paper's flexibility is its biggest benefit and this flexiblity must not be sacrificed in new systems. The whole purpose of this exercise is to eliminate the constraints of the systems we have imposed on ourselves over the last decades.

EXISTING TECHNOLOGY IS SUFFICIENT

No need to delay the planning for a system which will allow you to work smarter. You can start with what you already have and grow over time. The process is evolutionary and today's technology provides sufficient capability to begin now.

Over the past several years we have seen a clear trend where printer capabilites and intelligence have increased while the prices have generally taken a nose dive. As users, we face a win win situation because this trend will continue: we will get more value for fewer dollars or Marks, Francs, etc.

Today we can see where full font printers which lack print flexibility are quickly being displaced by dot matrix printers. An increasing number of users need the print flexibility of dot matrix impact or non impact printers. The new printers can print a variety of font styles including special graphics images. The lack of software which can take full advantage of the features continues to limit wide spread use.

For those who want everything in one printer (or almost everything), the answer may be in the newer laser or ink jet printers. Speed, print quality, reliability, font flexibiltiy, graphics, color (ink jet only) are available today on a variety of printers. Trade-offs between the features will affect overall price and performance. A full evaluation of the application and printer capabilities is necessary before the best decision can be made. Though it may appear that the printers provide all the answers, its utimately up to us as users to exploit the available technology.

As systems become more powerful, they can support a greater number of users and associated peripheral devices. It is not unrealistic to view future CPUs as peripheral device controllers. Data processing will remain a function of these controllers but they must be sufficiently flexible to satisfy the requirements of a growing user base as well as meet the challenge of the dynamic work environment characteristic of the information explosion.

However, just because printers today can satisfy a wide range of printing demands and, systems can physically support more users than before, it does not automatically mean users are able to draw data from the system and present it as useful and immediately usable information.

What about the software? Software technology shows a similar rate of development that hardware has shown. Major advances have been made in data base management systems which allow users easy flexible access to the data while maintaining appropriate levels of data security. Other software packages are available to manipulate the data (be it numbers or textual data) quickly and easily. Data communications software has made steady progress in permitting non-like systems to pass information between themselves. Even with these improvements and other advances in the sophistication of user interfaces, the passing of data from one software subsystem to another may require a degree of user intervention which renders the process impractical. In such a case, the user is in effect constrained by the system - precisely what we want to avoid in our quest to work smarter.

While it may be impractical for an individual user to make the jump between software systems, it is imperative that companies make the effort. The software does allow the necessary crosstalk which turns raw data into useful and usable information. The next section, suggests ways to change from the old methods to the new methods.

QUANTIFIABLE RESULTS

We have seen that with available technology we have an opportunity to improve the effectiveness and efficiency among the many users of hardcopy output. The next step is to set down a quantifiable method to measure, evaluate, and implement new approaches to hardcopy output in the way we work.

The process in principle is not new; the tools to help you are new. The recommended process follows a 8 step procedure:

1. Assign responsibility to a person or group
2. Clearly state objectives and other expectations of the program
3. Identify target areas for the study
4. Collect data
5. Analyze and Implement
6. Collect data for feedback
7. Acknowledge results
8. Attack the next area

1. Responsibility

Success hangs heavily on the people affected by the changes. A close personal identification between the people and the problem they are solving is necessary. Committment through involvement is important to ensure the program's success.

2. Objectives

People need a clear target. It must be realistic. The objectives should not imply constraints. Leave as much room for creative ideas as possible. The "We've always done it this way!" attitude is absolutely wrong.

3. Target Areas

Pick those areas which will yield the highest return for the effort expended. You may want to break all possible applications into 3 groups: existing system applications, user applications (word processing, personal computer applications etc) and, new applications. By grouping, one can quickly see which applications will provide the most benefit to the most people.

Existing applications might include accounting, personnel, payroll, forecasting, budgeting, materials management, production control, order processing or any of a host of possible applications you may have. Be sure to include even the small speciality systems you may have written yourself. Typically, existing applications are stable and output is generated over regular time intervals. The data collected and information presented by these bread and butter applications are the basis for the work done and decisions made in the course of a business day.

User applications tend to be the tools that individuals use to simplify their day to day work. It may be something as basic as word processing, or it could extend to a forecasting system on a personal computer. Since the individual has sole control in such cases, it should be left up to the individual to determine how they can work better. We want to focus on global systems as they will yield the greatest return.

New applications as they come into the planning horizen should be subjected to the same evaluation as existing applications. The applications backlog will most certainly change after the work study is complete.

4. Collect Data

If mistakes are made here, the whole program is lost. COLLECT THE RIGHT DATA! Since we are talking about improving the use of hardcopy output in the company we should find out how hardcopy is used now. Its best to approach the whole process as objectively as possible. If a bias is shown early in the process, then the final systems which get implemented may not be the best of all possibilities.

Using the grouping of the target areas, we would want the collected data from existing applications to answer such questions as,

PRINTOUT/REPORTS

What is being produced?
How often are they produced?
How many full and partial copies are made?
How is the output handled?
How and where are they stored?
How many times is a single report accessed (over time too)?
How long is a report stored?

DATA

What is the orginal data source?
How many different sources contain a single data item
(is it on-line too?)
Why is a data item accessed?
Which data item is accessed?
How is that item keyed?
How often is that item accessed (over time)?
What additional processing was required before the right information
was available?

PEOPLE

Does the person using the report have access to the system?
Who needed the information (function is sufficient)?
How long did the search, process, act cycle take?
What action is taken afterwards (corrective, preventative,
standard processing)?

Several methods and sources can be used to collect the data. Each has its strengths and weaknesses depending on the information sought. Some suggestions...

1. System Management Records
2. Data Management Records and Reports
3. Inventory Listings
4. Flow Charts
5. Work Study Logs

Items 1,2,3 probably already exist in any variety of forms. Let's take a closer look at items 4 and 5.

Flow Charts

A diagram of the complete cycle of a report from data input through generation, use and storage helps put the whole process in perspective. This exercise could yield interesting insights and highlight gross inefficiencies and waste.

Work Study Logs

The purpose of work study logs is to provide a statistical base from which one can draw conclusions and make decisions on how to improve the current work flow.

Entries are made into work logs by time, action or interview. An example of a time based entry is when at random intervals (every 8-15 minutes) a person logs the activity being performed at that moment. Such a study would show how a person's total work day is broken down by activity. For those people who do not deal regularly with standard reports it may be interesting to study how a person uses hardcopy output as a support tool for their job. In this case we would study a persons job and how it uses hardcopy.

In most cases for what we want to learn, our best choice is the action based method. In this case, every time an action is performed (use of hardcopy) several details are noted that describe the action. In our case, the details would refer to the report, data item, extra processing done, length of time needed etc. Limits to the number of details required for any one log entry are necessary to keep the record tracking simple. It's easy to go overboard which can severely threaten the integrity of the study data.

When designing the log sheets, 3 points should be remembered:

1. Logging must be simple as not to disrupt normal work - employ the use of ticks, checks and, abbreviations as much a possible. The responses must be standardized. Long textual descriptions take too long to write and cannot be quantified in analysis. A clear indication of the usage is what we are looking for. Too much detail can confuse the issue.

2. Analysis of the data must be easy (for fast turnaround) - be creative. Try having the logging procedure actually generate a graphics image. Such a process would eliminate a major part of the post analysis work. For forms printing applications, you might want to use a blank form as the daily logging form.

3. Data collected must be meaningful - people won't log it if they think it is a waste. Do not bother asking for data you think you might need sometime. If you are not sure what you want, you are not ready to run the test. If in doubt, don't.

Some examples of logging forms follow at the end of this document. Keep in mind that the design of the log forms is determined by a number of factors including specialization of the report/output and people using them, data to be acquired, number of different types of output studied by one person and, the duration of the study period.

5. Analysis and Changes

The results of the study could indicate a variety of possible changes to the current way of working. Some of the major catagories might be

1. Discontinue certain reports - gives a great feeling of satisfaction.

2. Reformat reports - design reports with ergonomics in mind.
   - use the information on type of access to see which data items should be side by side or on top of each other.
   - Highlight the most frequently accessed items
   - Put the items used as keys where they are easy to spot when looking through a lot of pages (upper right corner).
   - Print small (4:1 reduction for the information that must be printed but is not accessed very often).
   - Print graphics instead of numbers when a 'feeling' is all that is needed. This can dramatically reduce the time spent post-processing a report.

3. Put selective reporting on-line - some data is needed but a paper form of the report may not be required. If the person has a terminal, it may be enough to have a screen report. INFORM serves the same type of function.

6. Feedback

The same data collection procedure used earlier should be repeated after the new system is installed 6-12 months. The direct comparision of usage rates will give an indication of the improvements achieved. Interviews are also important to get a subjective feel for how well the systems is accepted.

7. Aknowledge

Publicize the effort and gains made. This is a real reward for the team of people who did the study and helped make the program successful.

8. Next Key Area

No need to stop after the first success. Hardcopy usage is prevalent within the organization and there are plenty of opportunities for improvement.

TOTAL SYSTEMS APPROACH

In order to dramatically improve, develop, and maintain high quality hardcopy output, it is necessary to fit the hardcopy plans into the overall systems plans. The possible ramafications of a study as outlined earlier could extend to modifications of data communications plans, data management projects and ideas of personal computer networks.

The approach I have outlined here can be applied to various aspects of the company. Some of you may recognize it as an approach sometimes used by quality circles. The process is not a simple thing to do well but with planning and a willingness to make it work among all levels within the organization, it could yield astonding results.

The information explosion is here. Good luck.

REPORT ACCESS LOG

DIRECTIONS: 1) Attach this form to a copy of a report.
2) Note information on how you use report in appropriate columns.
3) KEYs are what you use to look up the ITEMs

KEY #1	KEY #2	ITEM #1	ITEM #2	PROCESSING	ACTION TAKEN
P/N: Part Number S/N: Serial Number O/N: Order Number D: Date	P/N: Part Number S/N: Serial Number O/N: Order Number R: Manufacturing Run	SD: Ship Date TN: Transport Number I: Inspection Code R: Manufacturing Run	SD: Ship Date TN: Transport Number I: Inspection Code R: Manufacturing Run	%: Calculation ": Copy G: Make Graphics T: Transcribe	P: Preventive action C: Corrective action

The purpose of this study is to determine how report formats can be improved to provide more information more quickly and accurately than current formats. Your suggestions are welcome. Suggestions implemented will be awarded with a dinner for two at a restrurant of your choice. Thank you.

MPE in an Oil Industry Environment

J Moran and MC Turnill
Information Technology Department, Britoil, Glasgow, UK

1. INTRODUCTION

BRITOIL installed its first HP3000 computer system in October 1978 and now has six machines which act as local processors, arranged in a network linked to a central IBM mainframe. This complex serves a company of 2,500 staff who are engaged in oil exploration and production. Over 250 terminals (mostly HP) are connected to a network of CASE Statistical Multiplexors providing a very wide range of applications to accountants, engineers, geologists, clerks and secretarial staff. The mainframe is used primarily for numerical simulation with heavy CPU requirements and database work, while the HP3000's run a very wide variety of applications varying from commercial and financial systems to technical modelling systems.

The strategy of local processors, first formulated in 1977 has proved to be very successful and the choice of HP equipment has proved to be popular with end users and development staff alike for its user friendliness.

In spite of the success we are now finding it difficult to maintain our local processors at the serviceability levels required by our end users largely because of software unreliability and difficulties in obtaining the appropriate level of software support from HP.

This paper reviews the strategy we have adopted, some of our successful applications, and the expectations of end users. The impact of service interruptions are examined, together with an analysis of current software problems. Finally, some suggestions are made for an improved level of service.

2. BRITOIL

BNOC was founded in 1976 as a national oil company with a dual role of exploration and production together with a national trading and monitoring role. In November last year, BNOC was 'privatised' by separating the oil exploration and production arm (BRITOIL), floating it on the stock market and leaving BNOC as a wholly owned government subsidiary.

Last year we produced 57.2 million barrels of oil and achieved pre-tax profits of £521 million (£109 m. after tax).

BRITOIL, in common with many other oil companies, has widely dispersed offices, with it's Headquarters in Glasgow (1,200 people) housed in some 10 separate buildings, awaiting the construction of a new office block. Aberdeen, 160 miles away, is the home of the operational departments, (1,200 people) housed in three separate offices also awaiting completion of a new office complex. In addition, a small senior management group is based in London.

BRITOIL uses contractors on a large scale and currently has some 400 contractors based at Linwood, near Glasgow, working on the new Clyde field. There are also terminals in construction yards at Methil and Cleveland, and also two offshore installations (Thistle and Beatrice). Staff mobility is necessarily high because of rapid expansion and the nature of the oil industry. These factors create the need for highly effective network capability.

3. INFORMATION TECHNOLOGY WITHIN BRITOIL

BRITOIL has recently created a unified IT department which has overall responsibility for co-ordinating the development of:

- use of computer technology
- onshore communications links
- office systems
- applications.

The goal of the department is to provide a unified long term strategy which will serve the information needs of all staff within the company. Currently this is being achieved in terms of facilities by:

- a central mainframe (IBM)	- for high performance processing - large database applications
- local processors (HP3000)	- for data collection and validation - interactive systems
- WANG word processors	- for secretarial work
- CASE DCX network	- to link any terminal to any computer system from any office

A single terminal policy has also been adopted, which states that we should provide any one user with just one terminal with that combination of functions which he requires.

Another goal for applications development is to make significant improvements in productivity by the use of new software tools such as RAPID, PROTOS, FOCUS etc. As a result of a concerted and successful drive, it is hoped that we will not write any new COBOL programs after the end of this year.

4. OFFSHORE PROJECTS

A typical North Sea platform takes 5-7 years from conception to production and costs over $2,000 million. These platforms are designed to withstand 100ft waves and 120mph winds, standing in up to 500ft of water. At peak production one platform will operate up to 50 wells which may extend up to 10,000ft in depth and distance from platform. Over 300 staff are required to operate a platform on 24hr/365 day basis. Typical production rates are 50,000 to 200,000 barrels/day yielding $1.5m to $6.5m a day.

Computer technology now plays a vital role in every stage in the life of a major off-shore project, from the initial exploration stages, using seismic surveys to the final production platform. Computer systems are used by nearly every department as a normal business tool which aids the user either in keeping track of information or by modelling either business decisions, or physical structures such as platforms or reservoirs. The benefits in productivity and accuracy, together with the pressure of large capital decisions and the need to keep the oil flowing makes these systems a vital part of the users' everyday work. Any loss of this service is now just as critical as the loss of power, or the loss of the telephone exchange.

5 USERS

We have divided our users and their systems into three broad categories, ie:

- 'workers'
- professional decision makers
- strategic management

5.1 'Workers'

Some examples are platform operators, maintenance staff, warehousemen, clerks (finance, personnel etc), construction staff and secretaries. These people are rarely trained in computer techniques, and typically require:

- input mechanisms which are simple to use
- terminals
- rapid response
- reporting which they can understand
- modified reporting to meet new needs on short timescales.

5.2 Professional Decision Makers

These will usually be professional staff who may well have been trained in some aspect of computer technology. Examples in this category are petroleum engineers, structural engineers, accountants, lawyers, personnel officers, administrators, some of whom will have a very high awareness of computers. Typical requirements are:

- meaningful model (or view) of their discipline
- batch and terminal facilities
- high use of CPU power
- rapid ad-hoc reporting
- complex graphics

These users have traditionally put up with complex interfaces to their computer systems, which today is becoming much less acceptable.

5.3 Strategic Managers

This group makes strategic decisions which affect:

- policy
- capital investment
- technical development
- staffing
- environment.

Although difficult to generalise, it is likely that a senior manager may have no experience in computing. Typical requirements are:

- terminal based
- easy access to wide range of summary data
- rapid response
- ad-hoc reports in short timescales
- graphics.

The key is to ensure simplicity of use. The senior manager cannot afford any complexity in operation, since it loses him time and he will abandon computer usage unless it helps him in decision making.

6. SOME APPLICATIONS

The first HP3000's were installed some two years before the IBM mainframe due to a number of complex factors. This resulted in much wider range of applications being developed for the HP machines than might otherwise have been the case. While the majority of the systems we have developed fall into the 'worker' category, we have also developed a number of technical modelling facilities in FORTRAN. Examples of both types are discussed below.

The workloads on each of our machines are generally determined more by location and balance rather than any attempt to separate work on a departmental basis. The six machines are used as follows:

HP3000/64	(HP1)	Glasgow (Cadogan House)	Production
HP3000/64	(HP2)	Glasgow (Cadogan House)	Development
HP3000/44	(HP4)	Glasgow (Centenary House)	Clyde Offshore Systems + CAD
HP3000/III	(HP20)}	Aberdeen (St Machar)	Mixed Production and Development includes ICE and MCS (see below)
HP3000/III	(HP21)}		
HP3000/III	(HP3)	Glasgow (Cadogan House)	Central Service (eg SLATE, Graphics) + System Programming.

6.1 Steel Control System

The main problems encountered in the early platforms were often excess or shortage of material (through lack of timely information) or late material delivery. In addition, there is now a statutory requirement for all steel plates to be index numbered. To attack these problems new systems have been developed which are based on interactive terminal systems to replace the earlier, ineffective, batch systems.

The systems adopted form a pattern of six feeder systems, all on HP3000, which exchange information with a central project co-ordination systems (Figure 1).

Each system is, however, self-contained and can operate in its own right. The terminals can be located anywhere with connections to the telephone system which allows siting not only in the company's HQ but also in construction sites, storage yards etc.

A central feature of the steel control system is the 'material' database, from which regular, on-demand exception reporting can be obtained. Details of all orders are entered 'on-line' directly so that the system operates in real time. Over 50,000 tons of steel are involved in a major platform. Each steel plate ordered is indexed by the tonne, grade and quality and in addition, has a unique serial number, with specification dimensions, with harth and cast numbers. This linking of plates to their related documentation is of vital importance to the systems success. The steel is shipped to a variety of storage yards in construction sites and the systems is designed to:

- prevent delays in construction schedules due to material shortages on individual sites
- minimise surpluses and hence costs
- maximise the use of steel off-cuts
- audit deliveries.

This system is built using a combination of VIEW, IMAGE, QUIZ, IBS and SKIPPER:

6.2 Manpower Control System

BRITOIL has a statutory obligation to produce a list of all staff who are offshore at any point in time. This must be accurate and contain personal details such as address, date of birth, next of kin etc.

This requirement is also linked to the need to allocate beds on offshore installations and to allocate places on offshore helicopter flights.

Two such systems are run - one for the Thistle complex and the other for Beatrice. This is achieved by logging information at various points, (figure 2).

- Travel Department in St Machar House, Aberdeen
- Logistics Department in Athenaeum House, Aberdeen
- Airports (Dyce, Sumburgh)
- Thistle/Beatrice landing pads.

The main reports produced are:

- contractors report	- all contractors listed offshore
- emergency worksheet	- full details of all personnel offshore
- personnel report	- details of all personnel on file
- occupation analysis	- details of skills offshore

and on-line enquiry facilities are also provided.

The key requirement is to ensure that if the HP3000 system fails, an alternative machine can immediately take over, since it is vital for all records to be up-to-date in an emergency situation. This is achieved by either DS links or magnetic tape transfer.

6.3 ICE - Inventory Control and Evaluation System

This system

- maintains current stock balances at all warehouses
- produces effective evaluation of inventory holdings and transaction movements
- provide enquiry facilities for inventory status
- links to corporate financial and other systems
- provides audit requirements.

This is a large suite coded in COBOL, using an IMAGE database, using QUIZ for reporting. VPLUS/3000 is used for terminal dialogue. Requires a fairly substantial database (700K sectors) and has links to four other key systems. Access is made by Materials and Finance Departments (eight terminals) and by Warehouse at Peterhead (four terminals).

6.4 Corporate Financial Model

This system is for professional management accountants.

UK oil tax is extremely complex, now requiring four interdependant layers, resulting in taxation at the 80-90% level. These taxes are usually altered every year in some major way, and such changes can critically affect new investment or development decisions. To make such decisions requires a corporate financial model, which includes all current projects and all tax models both past and present. The answers are critically dependant on assumptions made about inflation, oil price and exchange rates which may also be varied as part of the simulation process. The need is for a terminal interface to pose the questions, although the calculations themselves do not need to be performed in real time.

This system is built in combination of FORTRAN and FCS - a financial modelling language. The stack limitations made this a difficult system to implement.

6.5 CAD - DRAGON

This system written in FORTRAN by COMPEDA (now owned by PRIME) is implemented on HP3000/44 and provides a satisfactory design service for up to six professional users. The 16 bit architecture again poses limitations for this type of system. In addition, the HP FORTRAN compiler proved woefully inadequate to cope with symbol table sizes required by this application.

7. SOFTWARE TOOLS

In order to improve productivity use is made of many different software tools either from HP or third parties. Currently we use:

(a) INPUT

VPLUS/3000

(b) DATABASE

IMAGE

(c) CODING

RAPID
PROTOS
COBOL
FORTRAN
FCS
OPTICALC

(d) OUTPUT

QUIZ
COBOL
FORTRAN
VIEWDATA

(e) GRAPHICS

DSG
DI/3000 (from Precision Visuals)
HPDRAW
EASYCHART

(f) WORD PROCESSING/ OFFICE

SLATE

(g) COMMUNICATIONS

MRJE Link to IBM 3033N
DS/3000
DS/1000
IMF

We also have laser printers in Glasgow and Aberdeen which have revolutionised the production of output. This involves using laser software and also a specially adapted version of DSPRINT (obtained from HP on a no support basis). Most of IBM output is also directed to these laser printers using either MRJE or tape, leaving only one printer directly on the IBM.

8. USER SERVICE

Most of our systems are well liked by the users, for their ease of use (helped by VPLUS etc) and by development staff using many of the latest software tools on the market. However, certain systems can now be critically affected by system outage (eg, Manpower Control System, Inventory System, Corporate Finance System, Document Control System for Capital Projects).

As the company has developed, these systems now handle either vital information or data volumes which cannot be manually re-processed. Any significant system outage can have major impact, for example, non-availability of the MCS system could have potentially very serious consequences in an off-shore emergency. With volume systems (eg, ICE, Finance etc) significant outage causes user frustration and much overtime to catch up backlogs as well as wrestling with databases which may have become corrupted or out of date.

To a contractor on a big project, long computer delays are often intolerable and could trigger penalty costs to BRITOIL, as well as project slippage.

9. MPE PROBLEMS

Currently we are suffering far too many MPE system interruptions. Figure 3 shows total number of problems affecting at least one live system, broken down into four priorities, of which the first level implies serious interruption to service and level 4 is low priority. While this has improved since IMIT was implemented last October, we are still experiencing around 22 per month (on six machines) of which about three cause serious interruptions to service (Priority 1). Of these 22 we are reporting about four new SRs to HP every month (Figure 4). The average response time for HP to acknowledge these is currently around seven working days (figure 5) having improved from 17 at year end.

The average time for a reply is around 20 working days (figure 6),which for top priority problems is far too long. Of 22 problems raised each month typically

- 7 remain permanently unresolved
- 8 are worked around (ie circumvented)
- 7 are cleared (Figure 7). Many are noted as 'cleared' in subsequent MPE releases, which we are not in a position to implement as discussed below.

This rate of incidents causes two major problems:

- interruptions to service - with delays to user activity
- frustration among systems team - who feel they are on a constant treadmill of unresolved problems.

Thus, while we have excellent local support from South Queensferry, we feel that central software support services do not meet our needs as a large customer, running a wide variety of applications on the HP3000s.

10. MPE SUPPORT

MPE support works formally as shown in Figure 6. The rationale is that if the problem has never happened before it can be regarded as a "one-off funny" which can be ignored. Such a philosophy is not acceptable in our environment since it leaves so many unresolved problems. Worse, until recently (May 1983), such occurrences were not even logged by HP centrally in the UK. This meant that some problems were never addressed and others were not being correlated across sites.

To improve our understanding of the problems BRITOIL usually prints all dumps, and works with local support staff to tackle the problems.

We see seven key problems with MPE support from our viewpoint:-

1 aimed at disaster situations only, ie whole machine down. Key user applications which are down are not counted.

2 lack of prioritisation by HP on problem resolution. No understanding with customer of relative importance to customer of each failure.

3 one-off funnies "thrown away" some of which may be very serious

4 if a problem is not reproducible it is not a problem

5 no formality in documenting and issuing patches for each release ie, HP do not know state of the system our machines. We have had duplicated patches or clashing patches issued.

6 documentation of MPE is usually far too late. (Recently received IMAGE manual relating to release over one year ago).

7 PICS only operates on English working days (regional holidays vary) from 9am - 5pm. Responses have improved over the last year but are still highly variable (see figure 6) and very poor continuity is maintained from day-to-day due to staff rotation.

No out-of-hours support to cover emergency situations, (eg corrupt IMAGE databases etc.) as perceived by customer.

HP ability to provide fixes to problems to MPE within the UK appears to be severely hampered by the lack of an agreed overall release policy which mean that a very high percentage of fixes are 'in the next release' which the customer may not be in a position to implement.

11. MPE RELEASES

Each new MPE release has traditionally contained a very high level of change both to base software and to the middleware products which sit on top (eg, compilers, database, products etc). Past experience has shown us that in our environment we cannot implement such large changes overnight and it is vital that live applications should be re-tested before going into production. This was vividly demonstrated when we installed IMIT at the same time as model 64's. Numerous problems were encountered, especially with the ATP which forced the schedule to become very long. The planned time to implement a new release on six machines is now estimated to be around 50 man/days over 2.5 months, and hence needs to be pre-planned up to one year in advance. On this basis we cannot contemplate upgrading our machines in this way more than once a year and for this reason we skipped the CIPER release (with HP's agreement).

11.1 QMIT

We have produced several project plans which have been constantly altered since HP have failed to meet all planned release dates for QMIT.

The problems we have faced are graphically illustrated below:

1 Originally scheduled for November/December 1982.

2 In February 1983, stated as due in mid March. Our first proper plan drawn up to start in April 1983.

3 Documentation available at end of March. (ie, SSB, communicator and some manual updates).

4 Arrived on 25 April 1983

5 Followed plan (shown in Figure 7) starting on 28 April.

6 On 19 May, after 20 man/days effort expended and first major machine about to be upgraded, HP informed us of significant problems and advised us to stop.

7 On 20 May we agreed with HP and were informed of new issue of QMIT in mid-June.

8 New plan produced.

9 On 5 July new release is 3 weeks late and no information is available.

10 On 15 July informed (verbally) that it will arrive in 'one week's time'.

11 New plan produced.

12 On 22 July informed that although the release is within the UK it is withdrawn and a new release called QMIT-DELTA will be released in mid-August. No further plans possible at time of writing.

Meanwhile our backlog of IMIT problems "resolved in QMIT" grows even longer and the incident rate is increasing with load (as in the past).

Although it is known that MPEV will be the next release "in November/December" we can obtain no clear information as to release dates. Neither is it clear whether the three stage release strategy, announced last spring, of main release followed by two delta (bug fixing) releases will be followed.

Another important need is for advance strategic information about the contents of the next release. At present we receive the communicator only at release time which is too late for us to determine the impact on running systems.

12. A WAY FORWARD

We have set out below a number of proposals which we believe would significantly improve the software support service.

1 For HP to create an MPE Release Strategy which is adhered to and clearly understood by customer and company.

2 For Release Dates to be met. These first two will enable the customer to plan ahead and avoid jeopardising production loads.

3 Reduce the Number of Releases. One a year is the maximum that a large organisation such as Britoil can consider.

4 Separation of MPE Release from Products. Decoupling products (eg. DSG, compilers, etc) from MPE release could have major advantages to customers in reducing level of simultaneus change.

5 To Improve Documentation. At least six weeks prior to release (to the customer) preliminary (outline) documentation should be available so that customer can assess impact on running workload. The SSB also needs a complete re-vamp - it is poorly organised, poorly indexed and does not contain information on unresolved problems.

6 Establishment of International Problem Database similar to other manufacturers, so that all problems (whether first occurance or not) are logged and all fixes are known on world-wide basis. This would enable SSB to be kept up-to-date and improve confidence between customer and company

7 To Improve Load Testing of MPE prior to release. We often experience "first occurences" because of sheer range of applications we run. This would help to avoid some of the QMIT release problems.

8 Agreed Priorisation of Customer Problems so that both HP and customer understand the relative importance of reported problems. At present only 'machine down' seems to have any impact.

9 An Improved PICS Service with more consistent responses, better continuity and an optional 'out of hours' service available.

10 Improved TSE support, especially for large customers so that better working relationships can be established.

11 Introduction of proper control mechanisms on patch release so that state of every system can be understood, abandoning binary fixes and turning to source fixes on regional basis.

12 Acceptance that a site can be critical if key applications have failed and cannot be restarted. This means defining the word 'critical' and the 'escalation' procedures which follow.

13 Allow customers systems staff to be able to read dumps (at diagnostic level only) to determine area of problem. Fix level should remain only with HP.

14 Enhancement requests should receive replies and be available for all customers to see.

15 MPE support should rest with SE organisation rather than CE.

Several of these ideas were mentioned in Roy Clifton's paper at Montreal but none have yet appeared to have any affect on support available to us.

13. CONCLUSIONS

These are resoluble problems. Let me stress that our view of HP hardware and software as products is good and they can and do provide excellent solutions for our end users. As the products mature it is however necessary for the software support organisation to mature at the same time. To achieve this requires the same level of commitment as is given to hardware by the CE organisation.

14. ACKNOWLEDGEMENTS

The authors wish to thank Britoil for permission to publish this paper. Numerous people within the Information Technology Department contributed to the paper, but particular thanks are due to DL Sheldrake, MD Vickers, S Owen and J McGrath.

FIGURE 1

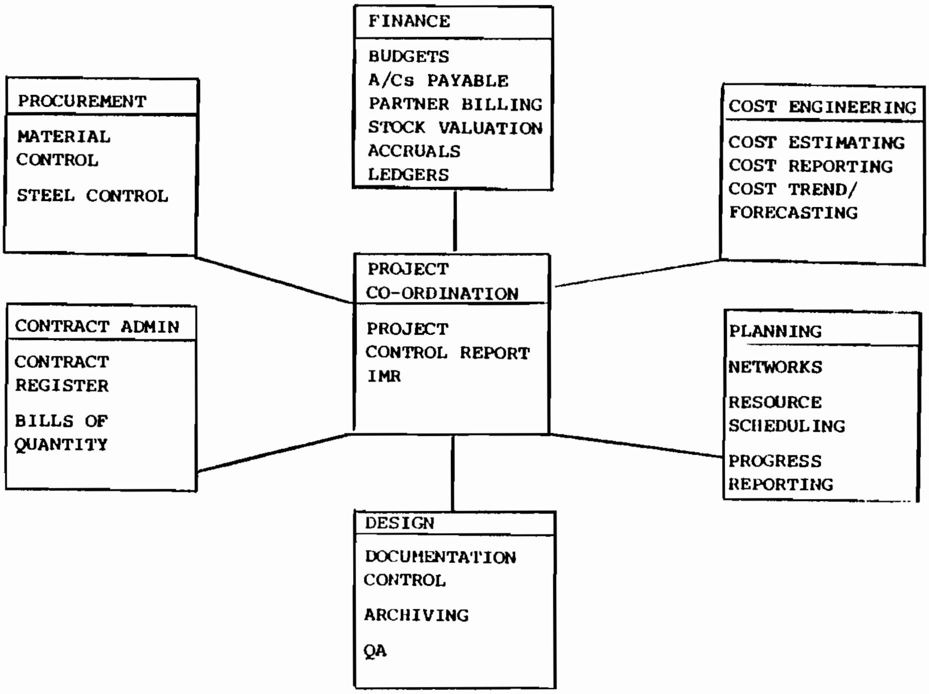

FINANCE
BUDGETS
A/Cs PAYABLE
PARTNER BILLING
STOCK VALUATION
ACCRUALS
LEDGERS
PROCUREMENT
MATERIAL CONTROL
STEEL CONTROL
COST ENGINEERING
COST ESTIMATING
COST REPORTING
COST TREND/ FORECASTING
PROJECT CO-ORDINATION
PROJECT CONTROL REPORT
IMR
CONTRACT ADMIN
CONTRACT REGISTER
BILLS OF QUANTITY
PLANNING
NETWORKS
RESOURCE SCHEDULING
PROGRESS REPORTING
DESIGN
DOCUMENTATION CONTROL
ARCHIVING
QA

FIGURE 2

MANPOWER CONTROL SYSTEM

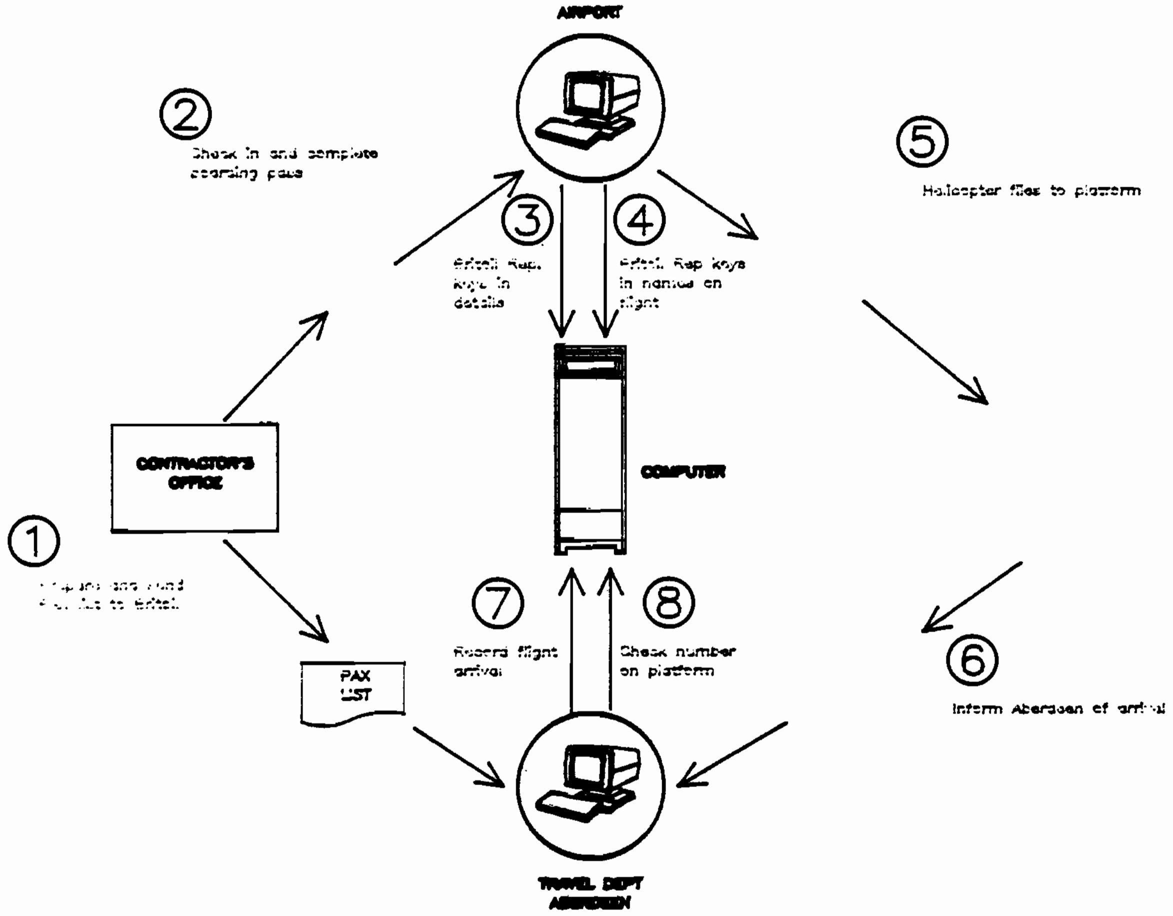

AIRPORT
① Prepares and sends
PAX list to Shell
CONTRACTOR'S OFFICE
PAX LIST
② Check in and complete
boarding pass
③ Shell Rep.
keys in
details
④ Shell Rep keys
in names on
flight
COMPUTER
⑤ Helicopter flies to platform
⑥ Inform Aberdeen of arrival
⑦ Record flight
arrival
⑧ Check number
on platform
TRAVEL DEPT
ABERDEEN

FIGURE 3

TOTAL PROBLEMS AND NO OF PRIORITY ONES

MAY 82 / JUNE 83

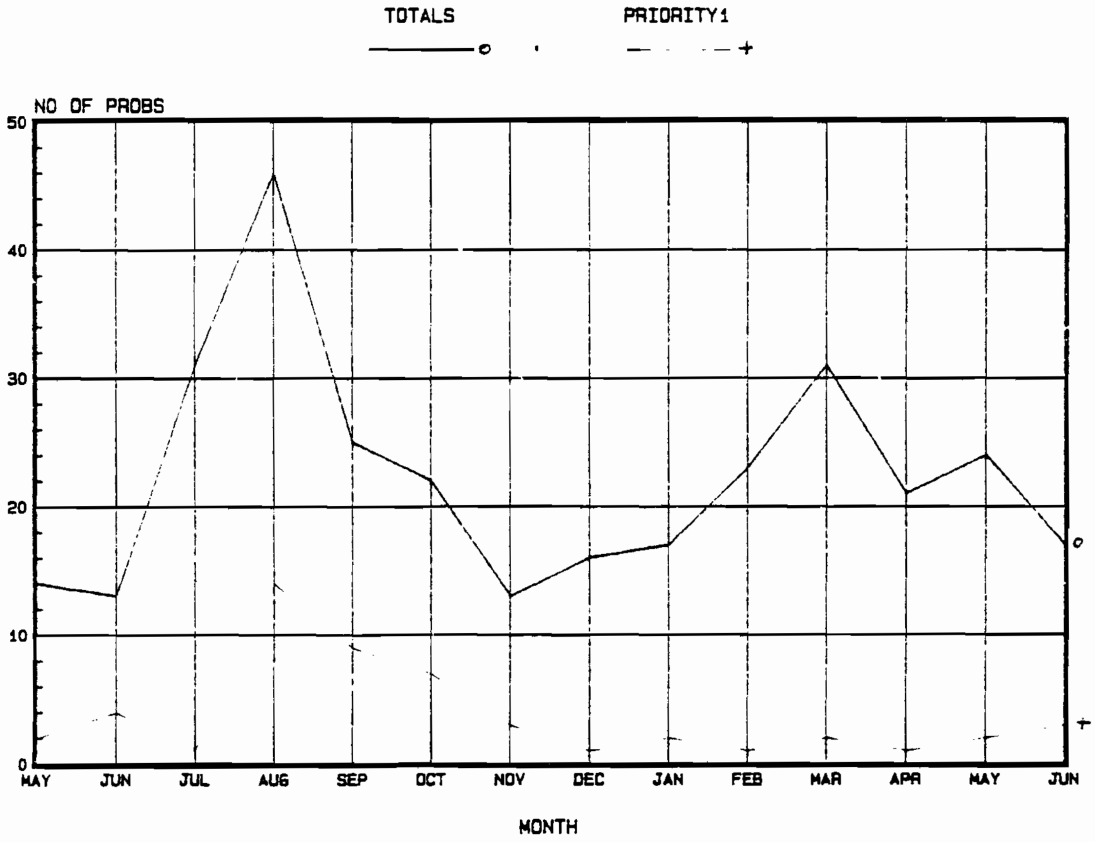


FIGURE 4

NUMBER OF SR'S TO HP PER MONTH

MAY 82 / JUNE 83

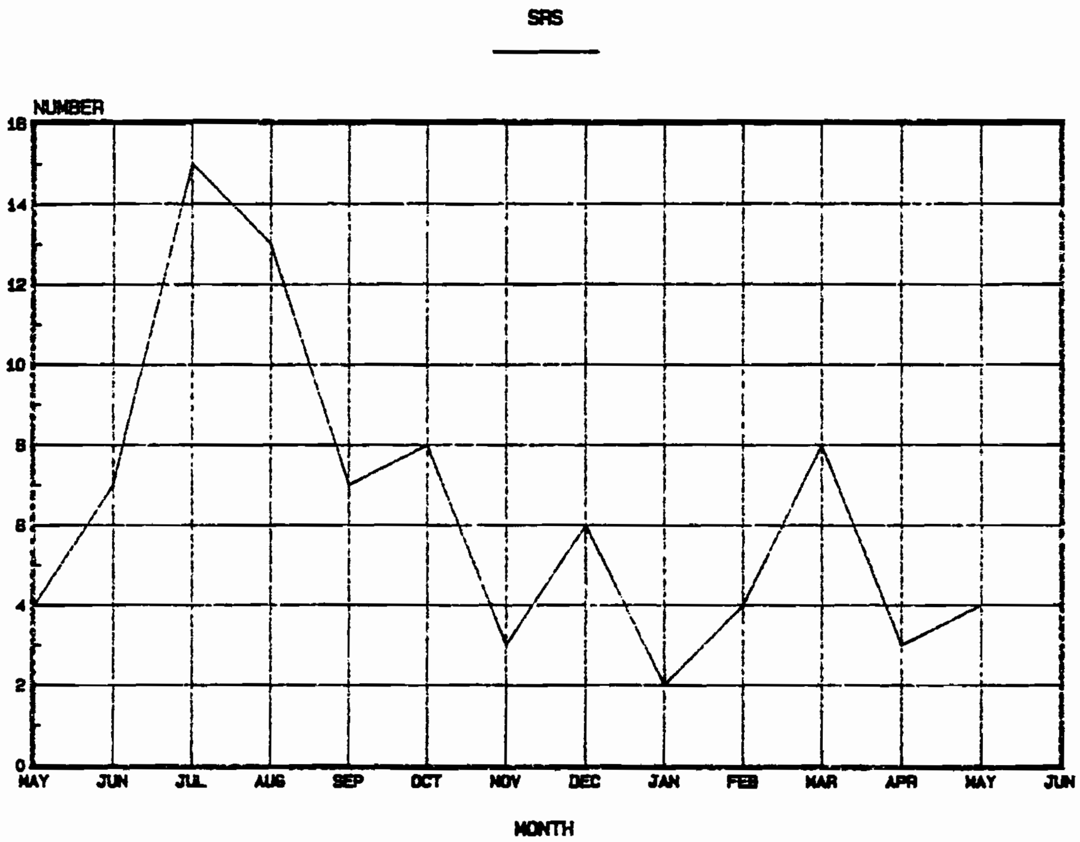
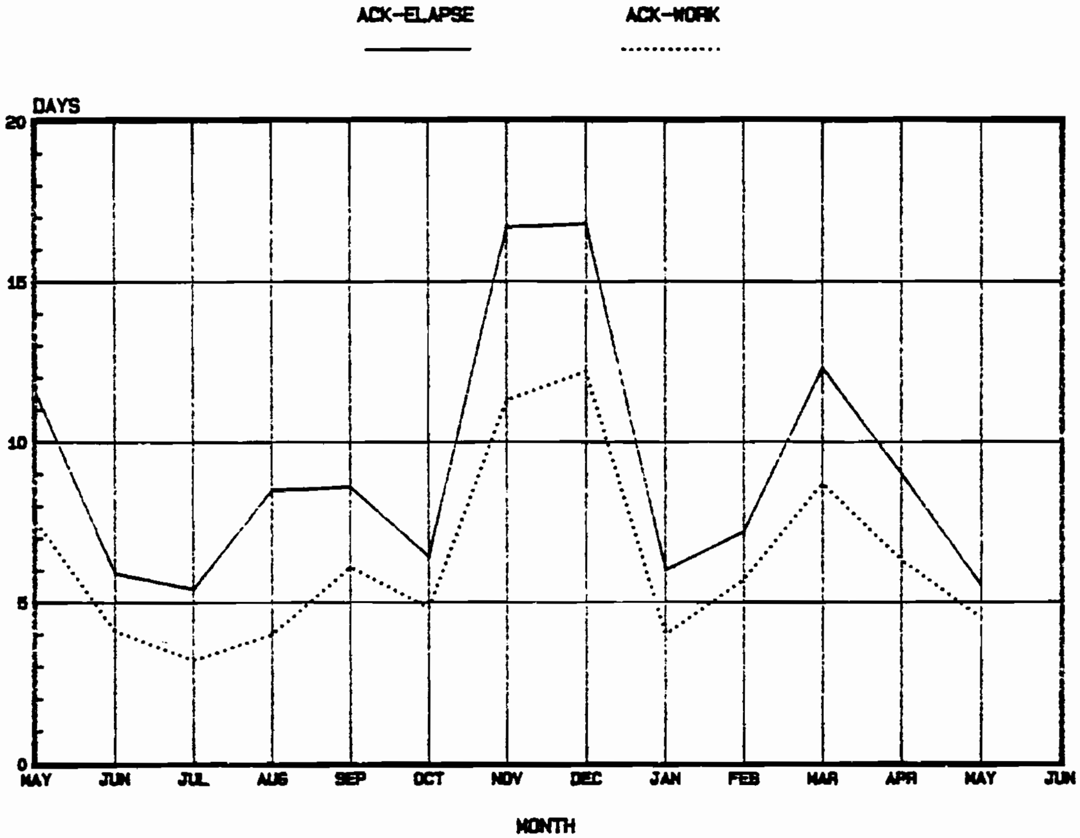


FIGURE 5

AVERAGE TIME FOR HP TO ACKNOWLEDGE AN SR

MAY 82 / JUNE 83



DIFFERENCE IS BETWEEN ELAPSED AND WORKING DAYS

FIGURE 6

AVERAGE TIME FOR HP TO REPLY TO AN SR

MAY 82 / JUNE 83

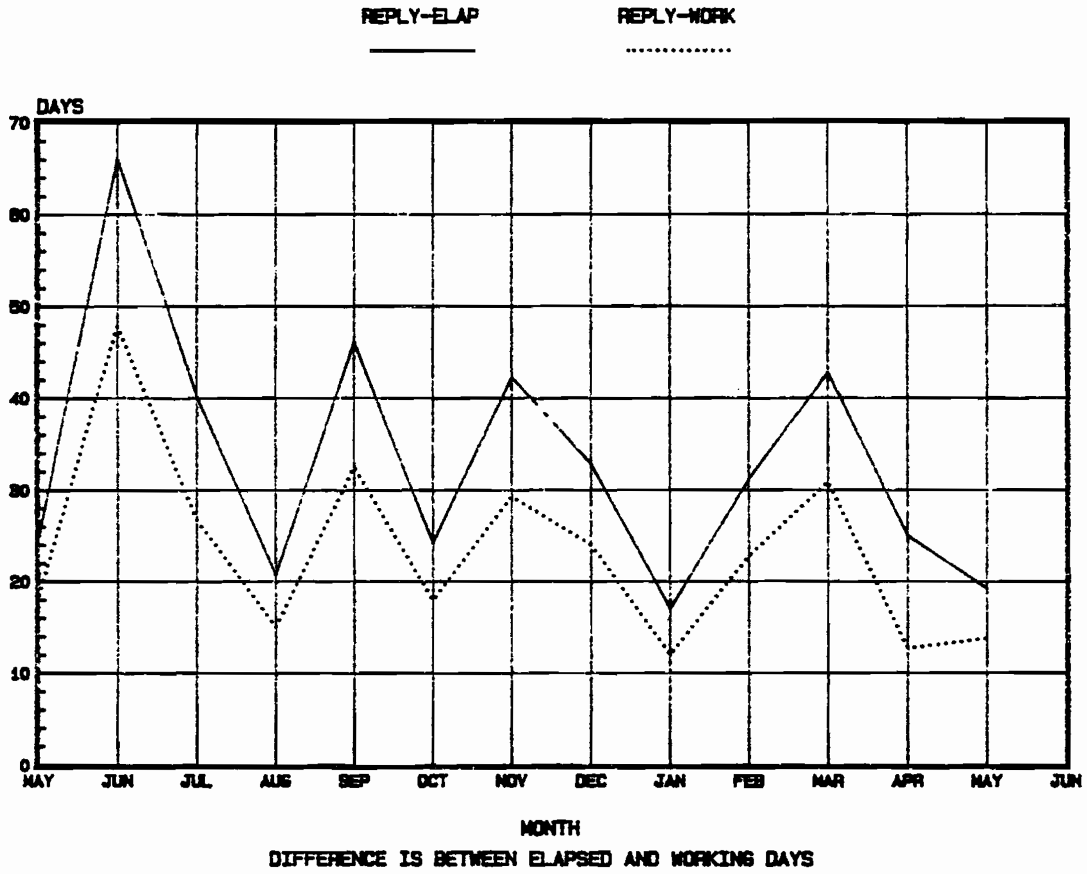


FIGURE 7

PROBLEM RESOLUTION STATISTICS

MAY 82 / JUNE 83

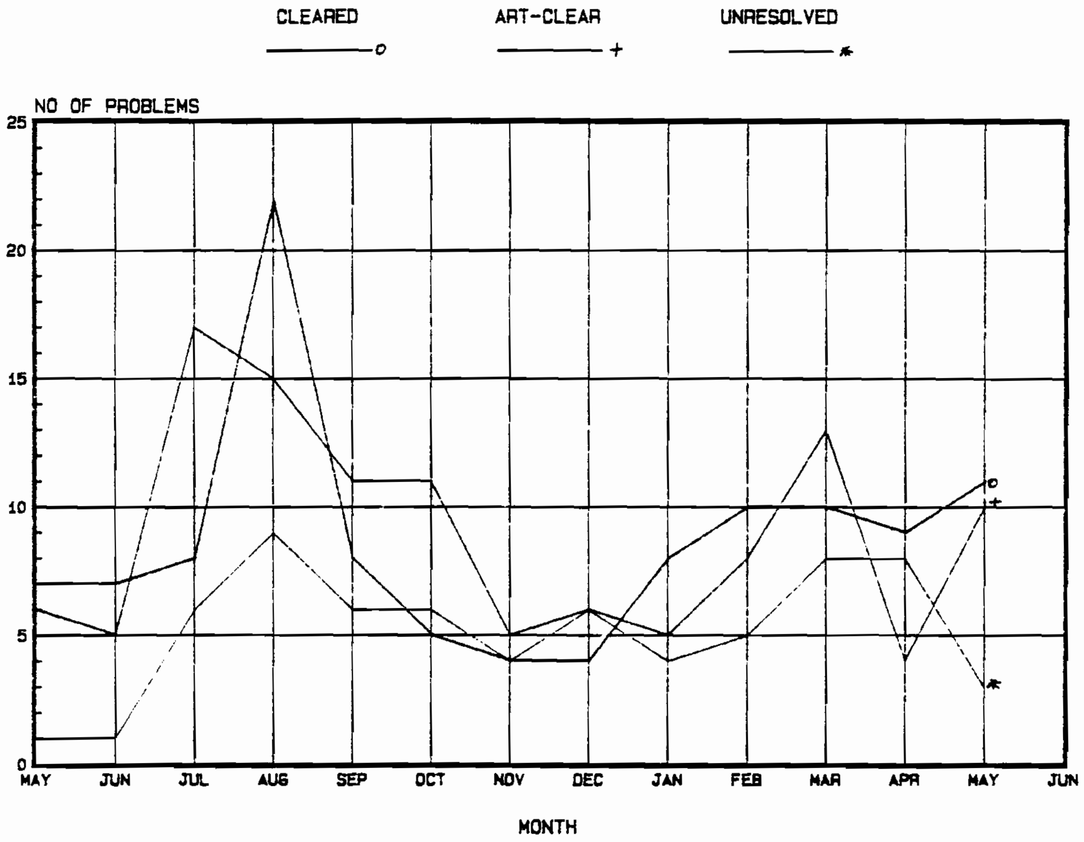
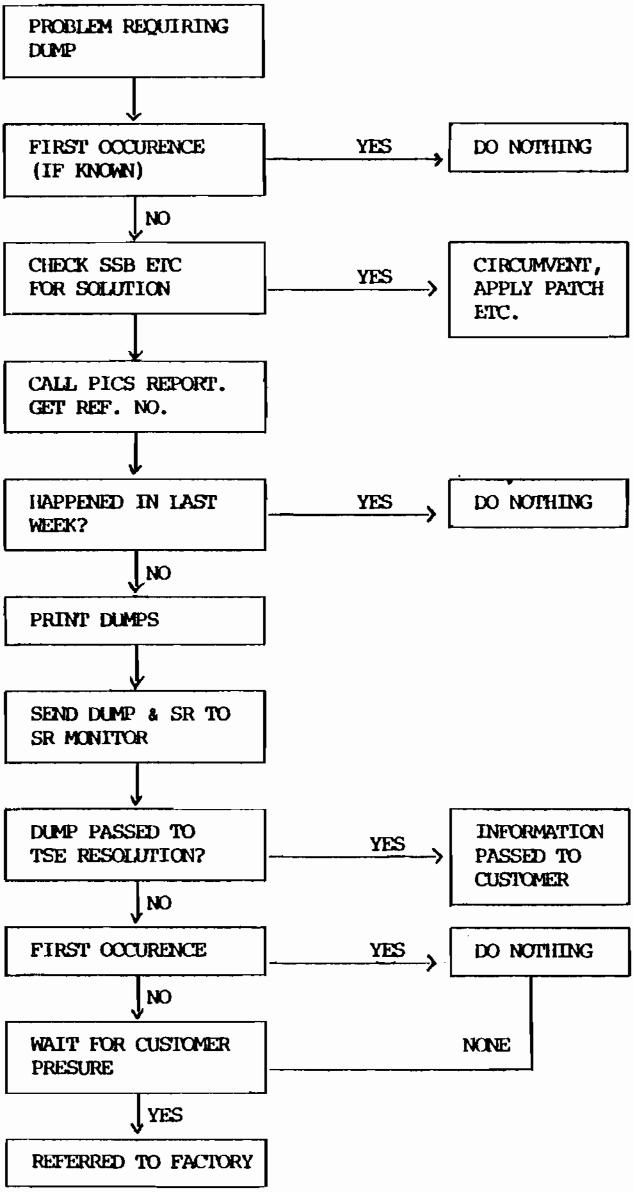


FIGURE 8

MPE SUPPORT PROCEDURE


PROBLEM REQUIRING DUMP
FIRST OCCURENCE (IF KNOWN)
YES
DO NOTHING
NO
CHECK SSB ETC FOR SOLUTION
YES
CIRCUMVENT, APPLY PATCH ETC.
CALL PICS REPORT. GET REF. NO.
HAPPENED IN LAST WEEK?
YES
DO NOTHING
NO
PRINT DUMPS
SEND DUMP & SR TO SR MONITOR
DUMP PASSED TO TSE RESOLUTION?
YES
INFORMATION PASSED TO CUSTOMER
NO
FIRST OCCURENCE
YES
DO NOTHING
NO
WAIT FOR CUSTOMER PRESURE
NONE
YES
REFERRED TO FACTORY

ON-LINE Support: An Inside Look
or
The Buck Stops Here (almost)

Karen Biglarderi
Nancy Ofslager
MPD On-Line Support

This paper is designed to give the reader an idea of the ways in which the field support organization and the factory interface in order to disperse product information, answer technical questions and resolve customer problems. In most cases, the primary interface between the customer and Hewlett-Packard is the System Engineer. Customers may be unaware of the large number of resources that can be called into play in the event of a serious problem. Thus, this paper shall also attempt to explain the process through which Service Requests are classified, questions are answered, customer problems are resolved, product decisions are reached, and the resources involved in each of these procedures.

As an On-line Support Engineer in the Manufacturing Productivity Division (formerly of the Information Networks Division), I constantly interface with System Engineers in order to resolve customer problems and relay information regarding the products which I support. Because the products which I support (VPLUS/3000, EDIT/3000, Datacap Intrinsics and DEL) are software products, this paper shall deal with the procedures and actions of an On-line Support group dealing with software products.

Basically, the job of On-line Support is to be the System Engineer's interface to "the factory". Among the responsibilities of an On-line Support (Marketing) Engineer are: answering questions posed by System Engineers (SE's), reviewing Service Requests (SR's), reviewing manual updates and rewrites, testing software, attending product team meetings (where a product's strategy is determined), becoming involved as a factory contact for all problem sites, writing technical articles for publications such as The Communicator and Support Update, becoming involved in the design of new products and occasionally reviewing, writing or teaching SE training courses.

There are many different On-line Support groups which deal with HP3000 products at Hewlett-Packard. Each group specializes in a different set of products, such as: languages, terminals, office products, business application tools, operating systems and databases. Most of these On-line Support groups are located in various divisions, but perform similar functions.

Typically, there is one On-line Support Engineer for each active product such as VPLUS/3000, Pascal or Toolset and a team of support engineers for "products" such as MPE. Extremely new or active products such as RAPID/3000 or Image/Query have two On-line Engineers assigned. Each engineer is also assigned several less volatile products such as DEL, EDIT/3000 or FCOPY. A fully trained back-up is always available for an engineer's absent times.

Phone Calls

Perhaps the most substantial part of an On-line Support Engineer's job is answering questions presented by SE's, originating either from a PICS call, or from SE's own accounts. Most other work and responsibilities are performed between telephone calls. Questions received range from "I remember hearing once that..." to "I can't find out how to..." to "My customer is having trouble..." to "Does ... have any plans to ...". In the month of April, Business Application Tools On-line Support, which encompasses VPLUS, Image/Query, Rapid, KSAM, Editor, Fcopy, Sort-Merge, DEL, HP300 systems, and six support engineers, received 814 calls from SE's.

As far as a single product goes, VPLUS/3000 receives between ten and twenty phone calls a day. Some of these questions are very technical and require researching or experimentation to determine the answer. On-line Support's sources include source code, internal and external product specification documents, publications such as Support Update, the Software Status Bulletin and the Software Release Bulletin, product manuals, TWX's from other divisions, other On-line Engineers and of course the ultimate source, the Lab Engineer. Obviously the Software Development (lab) Engineer can not spend his or her valuable engineering time answering every one of these questions, so the On-line Support Engineers try to research and discover as many answers as possible.

In the Business Application Tools On-line Support Group, a telephone call procedural policy exists. This policy states that "calls should be returned and answered within four hours or the caller should be told that it will take longer, either when the question is received or within four hours". This policy is consistant with the customer's software service contracts, and was instituted so that SE's on PICS are able to return their telephone calls within four hours.

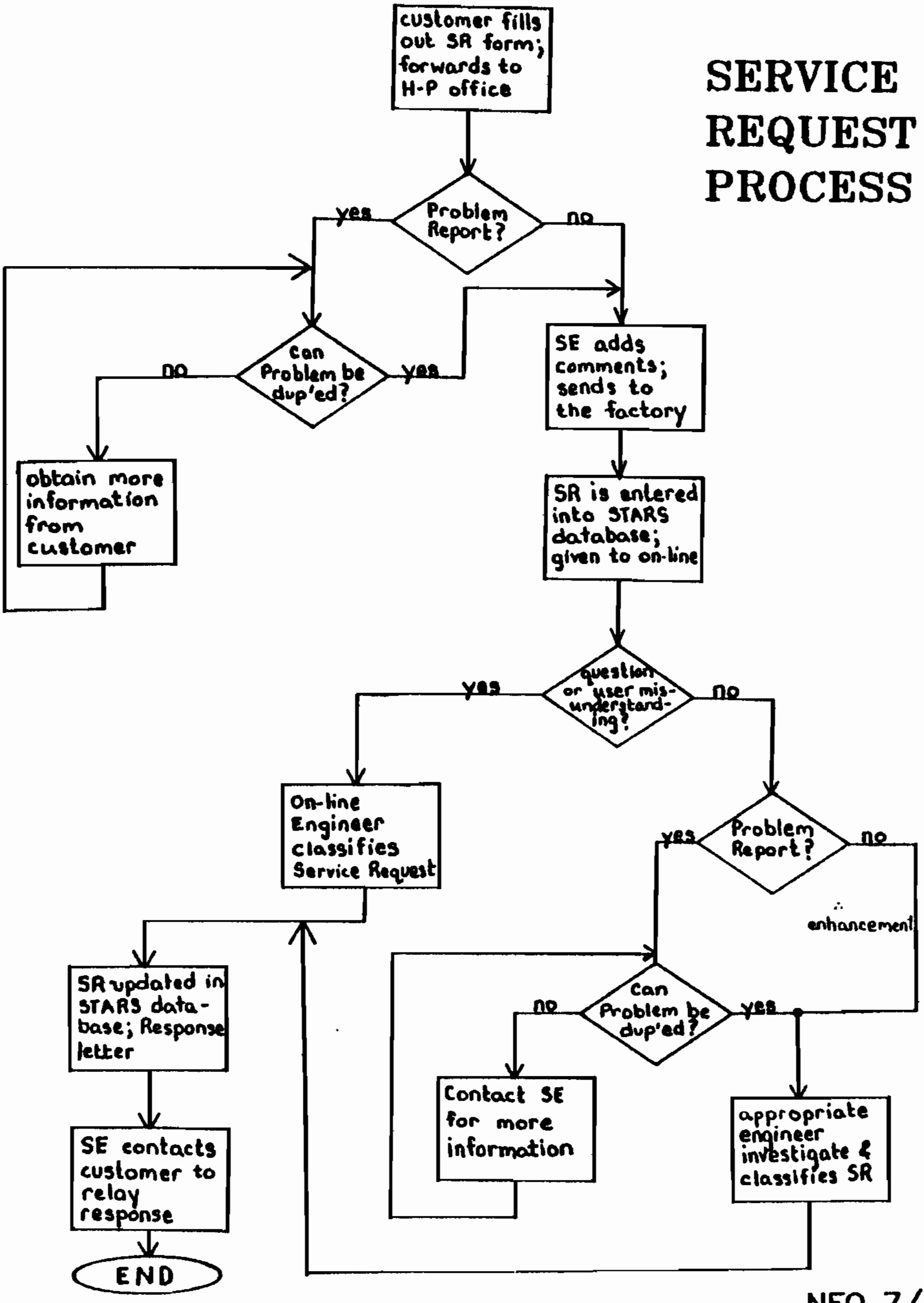
Most On-line Engineers also have their own policy on returning phone calls. Phone responses are generally made first to SE's who are at a customer site, then to SE's on PICS and then to SE's who have casual questions or inquires. In On-line Support we also have to consider time zone differences when we return calls. Many areas, such as Germany and Australia can only be reached at the very beginning of the day, so calls to these areas are sometimes returned first.

Service Requests

Another duty of an On-line Support Engineer is to become involved in the Service Request process. A Service Request (SR) is a report originated by a user of a Hewlett-Packard product to report a potential product problem, documentation error, receive an answer to a question, or request an (omitted feature) enhancement be added to the product. Contrary to the widely spread rumor, SR's do not fall into a bottomless pit or self-destruct when they reach the factory.

The life cycle of an SR generally follows the same pattern. After an SR form is filled out by the customer and forwarded to the local field office, an SE reviews the report, and attempts to reproduce any problems that may have been reported. If the SR reports a potential problem and the Systems Engineer can not duplicate the problem, he or she would call the customer for further information. Otherwise, the SE adds any comments they may have and forwards it to the factory in California (or wherever the product originates), with the files necessary to reproduce any problems.

SERVICE REQUEST PROCESS


customer fills out SR form; forwards to H-P office
Problem Report?
yes
no
Can Problem be dup'ed?
no
yes
obtain more information from customer
SE adds comments; sends to the factory
SR is entered into STARS database; given to on-line
question or user mis-understanding?
yes
no
On-line Engineer classifies Service Request
Problem Report?
yes
no
∴ enhancement
Can Problem be dup'ed?
no
yes
Contact SE for more information
appropriate engineer investigate & classifies SR
SR updated in STARS data-base; Response letter
SE contacts customer to relay response
END

NEO 7/83

Manufacturing Productivity Division

At the factory, the SR is entered into the Software Tracking and Reporting System (STARS) database. This database contains all "open" Service Requests and is used to generate the Software Status Bulletin, the Software Release Bulletin, response letters, action reports for all engineers involved in the SR process, and detailed reports regarding response time and classification breakdowns (%bugs, %enhancements). Many field offices now have the ability to submit and track a new Service Request on-line and add additional comments or information to existing Service Requests.

Once the Service Request is entered into STARS, the appropriate On-line Engineer is automatically assigned the responsibility of reviewing the report. At this point, duplicate problem reports are extracted and classified as such, and user misunderstanding reports are answered. Next, On-line Engineers attempt to duplicate any potential problem reports. We attempt to duplicate the problems, not because we do not believe the problem exists, but rather because we want to be able to demonstrate the problem to the Lab Engineer and have a test example available for his or her immediate investigation. Another reason why On-line Support must attempt to duplicate potential problems is because certain equipment which is necessary to reproduce the problem may not been available for the SE in his office. Duplicated problem reports,documentation errors and enhancement requests are then routed to the appropriate Software Engineer or Technical Writer, with any comments necessary to quickly resolve the report.

When the Service Request is classified (by On-line or the Lab), a response letter to the System Engineer is automatically generated, explaining the resolution of the report. If the SE disagrees with the response, they are able to telephone the On-line Engineer or reopen the SR. When the SR is ultimately resolved, the SE reports the findings to the customer.

To get an idea of the volume of Service Requests received by Hewlett-Packard, consider the Business Application Tools On-line group discussed previously. They received 165 Service Requests in April. Of this 165 requests approximately 44% were enhancement requests, 17% were new known problem reports, 13% were duplicate problem reports, 14% were documentation enhancements or errors, 5% were user misunderstandings, and 5% were classified as unable to duplicate.

Problem Escalation and Resolution

The resolution of customer problems is the number one concern of On-line Support. We like to become involved in problem sites as soon as possible in order to insure that customer problems are solved quickly. Frequently, On-line Support has encountered the problem before and knows of a work-around, patch, or version which does not have the problem. Sometimes, a "problem" is not a "problem" at all and a simple explanation can satisfy a very unhappy customer.

Occasionally, problems do arise which can not be solved by a PICS call and are too critical to wait for a response from a Service Request. For such problems, HP has established a formal process to insure rapid resolution of complex problems. It is called the Escalation Plan and its objectives are to provide resources to match the severity of a problem and lines of communication to insure the problem's timely resolution.

HEWLETT PACKARD

ESCALATION MANAGEMENT

PERSONNEL	STAGES	CONDITIONS	RESOURCE LEVEL	ACTION
Factory Managers Area Managers	HOT SITE	Remain until condition is corrected	ADVANCED Dedicated Factory Resources	FACTORY TAKES RESPONSIBILITY FOR TECHNICAL SOLUTION
On–line Engineers Lab Engineers	WARM SITE	Problem Continues another 4 hours	EXTENDED Factory Facilitated	FACTORY BECOMES INVOLVED ACTION PLAN IS UPDATED
District Manager Local Specialists Account Engineers	ALERT SITE	Critical Problem not diagnosed in 4 add. hrs.	FIELD Local	ACTION PLAN IS DEVELOPED AND APPROVED BY CUSTOMER

Critical problem not isolated in 4 hours

NEO 7/83

In those cases where the Customer Engineer (CE) or System Engineer at the customer site has difficulty isolating a problem, the support engineer will call a local specialist. Additionally, the District Manager will be contacted to assume the role of Site Problem Manager. These actions represent the beginning of the escalation process and will usually occur if a critical problem had not been isolated within four hours of the engineer initiating the diagnostic process at the customer site.

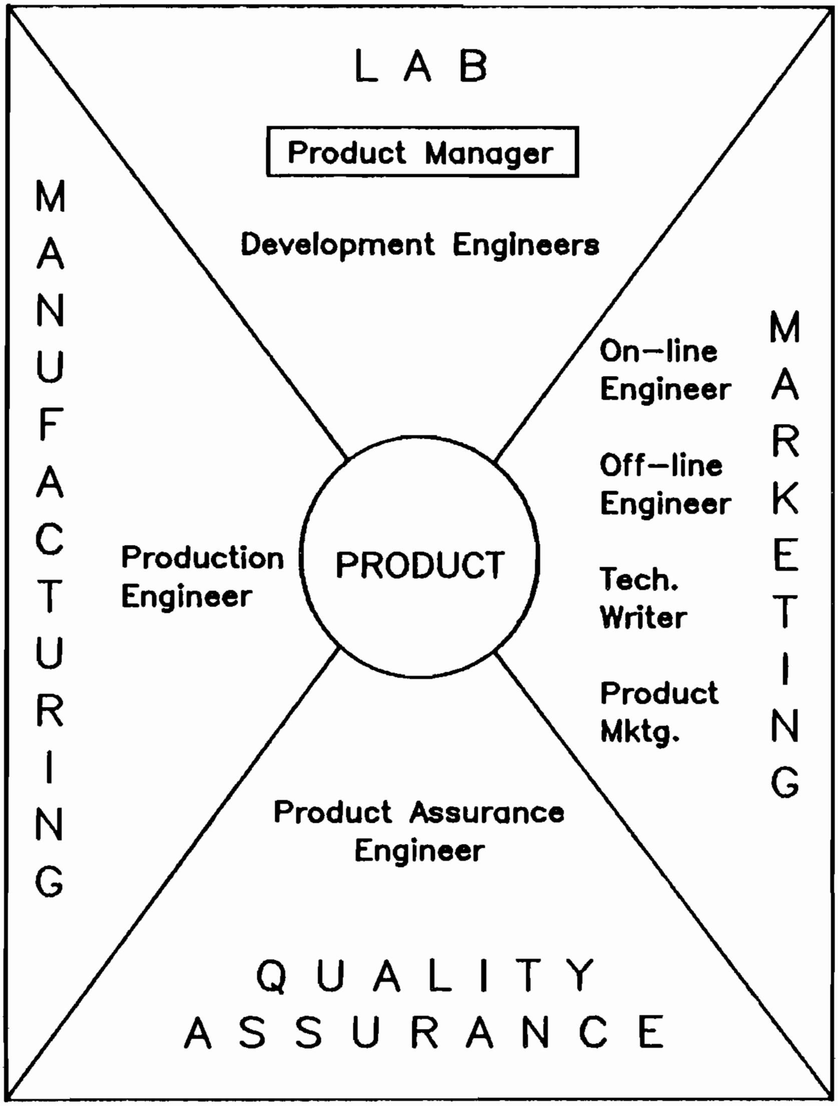
The Site Problem Manager will work with the CE and/or SE and the specialists involved to establish an action plan which includes a specific timetable for resolution and will present this to the customer for their approval. The specialized field engineer will consult with On-line Support to determine if this problem has been previously encountered. If the problem is an unknown one, On-line Support will offer suggestions on how to proceed to isolate and solve the problem. At this time the problem is considered to be at the "alert" stage.

If the problem continues to defy solution, the Site Problem Manager will implement procedures to escalate the site to the next level. This stage will generally be initiated for critical problems not diagnosed within eight hours, and is considered to be the "warm" stage.

At this stage, the Area Computer Manager, responsible for sales and support personnel in an area or country, coordinates all field resources necessary to resolve the problem. On-line Support becomes involved to coordinate the factory lab resources and communications with the field. When this stage is reached, On-line Support Engineers attempt to duplicate , the problem situation at the factory. This sometimes involves configuring one of the factory's stand-alone systems to match the customer's configuration. The Site Problem Manager will also draw up a revised action plan and timetable utilizing the new resources available and communicate this plan to the customer.

If more resources are necessary, the site will be further escalated to a "hot site" through the agreement of the Site Problem Manager and the Factory Support Manager. At this point, anyone involved at the factory end is in a "drop all other work and solve this problem" mode. Software Engineers responsible for the product in question work with the On-line Engineers to duplicate and determine the cause of the problem. In certain cases, such as when the problem can not be duplicated in the factory, an on-site visit by the Software (Lab) Engineers and/or On-line Engineers will take place.

Upon problem resolution, the customer site is placed in a Monitor status. All of the resources previously mobilized to resolve the problem remain on call in case the problem should recur. The local CE and SE will observe and evaluate the situation at the site to make sure that the problem is fully resolved, and the system is fully operational. The senior technical specialist will recommend the time period for observation and remain available for consulting. If the problem should recur during this time period, all resources will be recalled to develop a solution. After the problem is entirely resolved to the field's and the customer's satisfaction, the problem and its solution will undergo a final review.



LAB

Product Manager

Development Engineers

MANUFACTURING

Production Engineer

PRODUCT

MARKETING

On-line Engineer

Off-line Engineer

Tech. Writer

Product Mktg.

Product Assurance Engineer

QUALITY ASSURANCE

PRODUCT TEAM MEMBERS

NEO 7/83

The Hewlett-Packard Product Team

Representatives from all functional areas are essential to the successful release and support of any software product. To simplify this interface, between the lab and the world, product teams were established, with representatives from all of the support areas. The purpose of a product team is to provide filtered input to the lab team concerning the needs of the user community and evaluations of the product's ability to meet those needs. The product team is composed of representatives from the four basic areas: Marketing, Manufacturing, Lab and Product Assurance. This is not to say that the product team is composed of four members. It may require several people from any one area to adequately represent that function's interests and provide the necessary direction to the lab team.

As illustrated on the previous page,the members of the product team include a Product Manager, Development Engineers, On-line Engineers, Off-line Engineers, Technical Writers, Production Engineers, Product Assurance Engineers, and Product Marketing Engineers. The Product Manager is responsible for the overall investigation, development and maintenance of the product. The Marketing Engineers include On-line Engineers, Off-line Engineers (responsible for SE/CE training), Technical Writers (responsible for product manuals and Product Marketing Engineers (responsible for Marketing Surveys and product introduction). The Product Assurance Engineers must certify the quality of the product and the Production Engineers must make sure that the product is producible. Last, but not least, the Development Engineers are responsible for investigating, designing, creating and maintaining the product.

One of the most important aspects of the product teams at Hewlett-Packard is the collection of engineers which complete the team. As the engineers are all involved in different job functions, they are able to present a variety of views on all product decisions. The Software Engineers can determine the ease or difficulty of technical implementation, the On-line Engineers can determine customer/field considerations and Off-line Engineers and Technical Writers can determine supportability. All of these views are taken into consideration when making a product decision.

Another important aspect of the product teams is that they are the driving force of the product. Basically, the product team is solely responsible for the success of the product. Since decisions are made at this low level, by the engineers who most closely deal with the product, they tend to be rational, customer driven decisions. In this manner, On-line Engineers can heavily influence which features, enhancements and problems fixes will be implemented in a product. If a particular enhancement or workaround is frequently requested, then On-line Engineers can express that urgency to the lab team and request that the work be performed.

As you can see, On-line Support is where the buck stops, as far as the SE is concerned. It is the responsibility of the On-line Engineers to obtain a solution to a customer's question or problem by drawing in as many resources as necessary. Hopefully, this paper has given you a better idea of the inside workings of On-line Support, the processes through which problems are resolved, Service Requests are answered, and the resources which Hewlett-Packard has provided to obtain customer satisfaction.

MPE DISC CACHE : IN PERSPECTIVE

John R. Busch
Alan J. Kondoff

Members of the Technical Staff
Hewlett Packard Corporation
Computer Systems Division
19447 Pruneridge Avenue
Cupertino, California
95014

ABSTRACT

MPE Disc Caching is a major new performance product for the HP 3000 family of computers. MPE Disc Caching effectively utilizes the excess main memory and processor capacity of the high-end 3000 family members to eliminate a large portion of the disc access delays encountered in an uncached system. With disc caching, disc data is available at main memory rather than disc access delays with a probability that increases with main memory size. The MPE disc cache designers present an overview of the purpose of disc caching, its design approach, its advantages over the alternatives, and its impact on the system price/performance of the HP 3000 family.

... Why Disc Caching ? ...

A storage hierarchy can provide a cost effective system organization for computer systems. Each successive level of a storage hierarchy uses lower cost, but slower, memory components. By retaining frequently accessed code and data in the higher speed memories, the system can operate at speeds close to the access times of the fastest memories but at costs approaching those of the slowest memories. The price and performance of a computer system is dominated by the organization and management of its storage hierarchy.

Achievable system performance is a direct function of processor speed and utilization. Processor utilization is limited primarily by its waiting time caused by misses at various levels of the storage hierarchy. Thus, for optimal system price/performance, the processor speed and the capacity, speed, and management of the levels of the storage hierarchy must be matched. In order to fully utilize the processor capacity, the system must achieve a sufficiently high probability of finding data when referenced at the highest levels rather than having to go to the lower levels of the hierarchy.

When a low hit ratio at a certain level of the storage hierarchy is causing low processor utilization (thereby limiting achievable system performance), a number of alternatives exist to resolve the problem. These include improving the management policies of the levels, increasing the capacity of the level incurring the low hit rate, speeding up the access time of the next lower level of the hierarchy, and introducing a new level into the storage hierarchy. Cost and technology determine which alternative or combination of alternatives is optimal.

This paper focuses on the innovative solution to the main memory/disc bottleneck which MPE Disc Caching provides for the HP 3000 computer family.

The levels of the storage hierarchies of HP 3000 computer systems [1] range from:

1) Processor registers, to

2) Microcode store, to

3) Processor cache, to

4) Main memory, to

5) Discs, and to

6) Tapes

Each of these levels uses memory components which have a significantly lower cost-per-byte but a significantly longer access time.

The higher-end 3000 systems currently suffer from low processor utilization in some database and disc I/O intensive installations. This negatively impacts the growth capability for customers. The low processor utilization is due to processor idling while misses at the main memory level are being resolved from disc storage. Main memory capacities beyond a few megabytes may provide limited incremental improvement.

The 3000 R&D lab teams have been working to resolve this imbalance. MPE Disc Caching is the solution that HP has developed. MPE Disc Caching optimally exploits current cost and technology tradeoffs to eliminate the imbalance.

... What is MPE Disc Caching ? ...

MPE Disc Caching is an optional MPE subsystem which manages retrieval and replacement of disc "domains" or regions in excess main memory. It locates, moves, and replaces disc domains in main memory so that a significant portion of the references to disc storage can be resolved without incurring physical disc access delays.

The MPE Disc Caching policies are fully integrated with the MPE kernel, file system, and I/O system. This allows system performance to be optimized based on current resource availability and workload demand.

Ancillary to MPE Disc Caching, there is a set of external controls, measurement and simulation tools to allow users to manage and predict the operation of caching on their systems. Operator commands are available to enable or disable caching on a device basis, and to display general caching statistics for a device. Measurement tools have been enhanced to display relevant disc cache usage and performance statistics. A special simulation tool has been developed which analyzes disc access traces from HP 3000 systems and produces disc cache performance statistics for specified cache memory sizes. With this tool, main memory requirements to efficiently support disc caching can be predicted for existing HP 3000 installations.

... MPE Disc Cache Design Approach ...

The MPE kernel resource management mechanisms and strategies [2] provide an efficient, integrated approach to resource management. The MPE disc cache mechanisms and strategies are integrated with those of the kernel, and exploit the file system's knowledge of file structure and access method to enhance prefetch and replacement decisions for disc domains. Microcode assist is exploited to rapidly locate cached domains in main memory.

The kernel's main memory placement and replacement mechanisms are extended to handle cached disc domains in the same manner as segments. Thus, cached disc domains can be of variable size, fetched in parallel with other segments or cached disc domains, garbage collected, and replaced in an integrated manner with stacks, data segments, and code segments. The relative allocation of main memory between stack, data, code and cached disc domain objects is entirely dynamic, responding to the workload's current requirements and current memory availability.

When a request is made to access disc information, the list of currently cached disc domains of the specified device is searched using the *linked list search* instruction. If the requested disc domain is present in main memory, the data is moved between the process' data area and the cached copy of the requested disc domain. The process will continue executing without an interruption or process switch.

If the access request is a write and there is currently a write pending against the specified disc domain, the process' request is queued until the pending write is posted to disc. If the disc domain to be written is not currently cached, an available region of memory is obtained which is used to map the corresponding disc image - i.e., no fetch of the disc domain to be written is required. When the move effecting the write takes place from the process' data area to the cached image of the disc, a post to the disc is initiated. Only the portion of the cached disc image which is modified by the write is posted. After the move to the disc image is performed and the post to disc is initiated, the writing process is allowed to continue running without having to wait for the physical post update to complete. Disc integrity is insured within the operating system and subsystems through serial posting on a global basis of writes with posting order constraints. At the user level, wait-for-post can be specified on a file or system basis in place of the default no-wait-for-post.

When a request is made to read data which is not currently cached, the fetch strategy uses knowledge of the file blocking, extent structure, access method, and current memory loading to select the optimal size of disc domain to be fetched into memory. The fetch of the disc domain is initiated through the memory manager on the current process' stack without a process switch. The fetch is performed in an unblocked manner so that the requesting process or another process can run in parallel with the cache fetch from disc.

When a process completes referencing a cached disc domain in sequential mode, the domain is flushed immediately from main memory since it won't be needed again. In this way, memory utilization is improved over that achievable with the kernel's approximate least recently used (LRU) replacement algorithm.

With these mechanisms and strategies, MPE Disc Caching significantly reduces the traffic between the main memory and secondary disc storage and significantly reduces delays to read or write disc information. It does so in a manner which is superior to its alternatives when evaluated by cost, reliability, and performance measures. The advantages over its alternatives are discussed in the following section, followed by a discussion of its impact on 3000 system price/performance.

... Advantages Over Alternatives ...

There are several alternatives to MPE Disc Caching which could help address the high traffic rate and long delays between main and secondary disc storage on 3000 systems. Software alternatives focus on providing localized caching on a subsystem, file, or application basis. Hardware alternatives focus on reducing access time to disc storage or increasing the number of concurrent paths to disc storage.

The software alternatives are all localized and unresponsive to current memory loading conditions. The amount of memory devoted to the caching of a specific file or subsystem would likely be either excessive or insufficient at any given moment, depending on the current memory availability and the current workload demand and priority structure.

Access times can be improved by speeding up the discs, introducing a new level in the storage hierarchy between semiconductor main memory and moving head devices, or caching the discs with a peripheral cache. Access capacity improvements can be accomplished by adding more parallel paths to secondary store with more discs, controllers and channels.

Speeding up discs involves technology limitations and tradeoffs between performance, cost, and reliability. Access time improvements that compare with the order of magnitude improvement provided with MPE Disc Caching are unlikely at any cost, even with head-per-track approaches.

Introducing a new level in the storage hierarchy which exploits bubbles or CCDs as a gap filling technology has not proven to be cost effective when compared with exploiting high density semiconductor memory technology in very large main memories.

A peripheral disc cache [3],[4] can be built into the controller or disc or can be a separate system component which front-ends a selected subset of the disc subsystem. There are cost, performance, and reliability tradeoffs involved between each of these peripheral disc cache architecture alternatives. MPE global disc caching in the SPU's (system processor unit) is superior to the best achievable cache architecture by all three measures.

Evaluated with respect to cost, a peripheral cache requires additional power, cooling, cabinetry, and electronics as well as the caching memory which is required for disc caching in both peripherals and main memory.

Evaluated with respect to reliability, the MPE disc cache introduces no new hardware components into the system. The reliability is identical to the uncached system whereas any of the peripheral cache architectures necessarily degrade system reliabilty due to their introduction of hardware components. The software/firmware complexity of the two alternatives is roughly the same, so reliability degradation due to cache management is comparable. Since the MPE Disc Caching is built into MPE, it has the potential to evolve toward improving system reliability (e.g. automatic extent sparing on write failures). Also, the posting strategy of the peripheral cache is not integrated with the system posting strategy so that a consistent level of integrity is not guaranteed.

Evaluated with respect to performance, the MPE disc cache has clear advantages in access time, access rate, and cache memory utilization.

Access to cached disc domains is provided on the current process' stack with a combination of firmware and software. Not even a process switch is required. The access time is on the order of a very few ms which scales with processor speed. Access to a peripheral cache requires at minimum a trip through the I/O software and interrupt system, a process switch, plus the cache access time of the peripheral cache.

Achievable cache access rate of the main memory cache is roughly the inverse of the cache access time (several hundred accesses per second, scaling with processor speed). This rate is achievable because the main memory cache is a parallel server. When one process encounters a miss on cache, the processor can be applied to another process which can access the cache concurrently with the resolution of cache faults of other processes. This level of parallelism would be very difficult to achieve in any of the peripheral cache architectures. Also, the best case access of any peripheral cache architecture is several times that of the main memory cache. Therefore, even if full parallelism were achievable in a peripheral cache, the best achievable access rate can only be a small fraction of that achievable with the main memory cache approach.

Cache memory utilization of global SPU caching integrated with the MPE kernel is superior due to several factors. The amount of cache memory applied to a cached device is responsive to the current utilization of the device. The size of a prefetched disc domain is tailored to the structure of the data (e.g. the cache mechanisms fetch extents instead of fetching tracks which contain unrelated data or only pieces of the required extent). The replacement policy exploits operating system knowledge of access patterns (e.g. the policy flushes a cached disc domain from the cache memory after sequential reference and on file purging).

The traditional way of addressing the disc bottleneck on a disc-bound system has been to increase disc access capacity by adding more discs, controllers, and channels. This approach is very expensive, and and its improvement in secondary store access time and access capacity cannot match that provided by MPE Disc Caching.

... Impact On 3000 Family System Price / Performance ...

MPE Disc Caching provides a significant improvement in 3000 family system price/performance for a large class of workloads. With the higher-end systems and a large class of 3000 application environments, MPE Disc Caching effectively reduces the system cost required to achieve a given performance level and significantly increases the system performance which is achievable for a given cost.

Although MPE Disc Caching introduces increments in system cost due to the expense of the optional software product and additional main memory for caching, these increases are more than offset by the reduction in system cost due to the ability to exploit the very attractive $/Mbyte advantage of HP's large capacity discs without suffering a significant performance disadvantage.

In 3000 systems without disc caching, exploiting the roughly fourfold $/Mbyte advantage of the 793X discs over the 792X discs comes at a significant performance cost due to the effective fourfold reduction in disc access capacity caused by the reduction in concurrent disc servers. Since the disc subsystem is utilized at a much lower rate with MPE Disc Caching, this reduction in disc access capacity has a limited impact on system performance. Second-order cost impacts are also achieved in that disc maintenance costs are reduced due to fewer drives and lower utilization of the drives.

Achievable system performance for installations with excess processor capacity is significantly higher with MPE Disc Caching than that with uncached configurations.

System response times are reduced due to the reduction in queueing delays and service times for the resources required to complete transactions. Disc queue lengths and frequency of visits to the discs are reduced. The queueing delays and holding times of system and application locks are significantly reduced since the disc delay components of the holding times are reduced. Since disc and database queueing and service time delays dominate the response time in uncached systems for many HP 3000 workloads, the impact on system response time due to disc caching can be expected to be significant. System throughput is improved due to the reduced response times and reduced contention for resources.

As a workload grows, the upgrade to higher performance family members becomes a very attractive alternative from a price/performance perspective.

.. Conclusions ..

MPE global disc caching in the SPU provides an innovative solution to overcoming the huge access time gap between main memory and disc storage and the long database semaphore queueing delays caused by this gap. MPE global disc caching in the SPU provides a solution which exploits the current cost and performance tradeoffs of memory and processor technologies. The solution is superior to the alternatives when measured by cost, performance or reliability. MPE disc caching provides significant system price/performance improvements across the HP 3000 family as well as an attractive upgrade for existing HP 3000 installations.

REFERENCES

[1]. HODOR, Ken M and WOODWARD, MALCOLM E., "A High Performance Memory System With Growth Capability", Hewlett Packard Journal, March, 1982, pp 15-17.

[2]. BUSCH, JOHN R., "The MPE IV Kernel : History, Structure and Strategies", Proceedings of the HP 3000 International User's Group Conference, Orlando, April 27-May 1, 1981, F-9; reprinted in Journal Of The HP 3000 International Users Group, Incorporated, Vol. 5, No. 3,4, July/December 1982.

[3]. HUGERSHOFER, WILLI and SCHULTZ, BRUCE, "Cache Buffer For Disc Accelerates Minicomputer Performance", Electronics, Feb. 10, 1982, pp 155-159.

[4]. KRASTINS, ULDIS, "Cache Memories Quicken Access to Disk Data", Electronic Design, May 13, 1982, pp 41-44.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

OPTIMIZING SYSTEM PERFORMANCE

David S. Wertheim
Hewlett-Packard

WHY TRY TO OPTIMIZE SYSTEM PERFORMANCE?

The most often heard cry from users today is bigger, better and faster! In a nutshell, they want to be able to do more work in less time. Many people from the old school say "throw iron at them".[1] That may be necessary, then again it may not be necessary. You wouldn't go to a Dr. who after you told them you had heart pains suggested open heart surgery would you? Would you? Often times fine tuning can improve performance of existing systems and satisfy user requirements. We will examine the most important factors that effect computer system performance and discuss measurement and evaluation techniques that will lead to reduced user response times, and improved system throughput.

Reducing user response times and improved system throughput seem noble enough goals, but these two goals have caused more grey hair and ulcers for MIS directors and system managers alike. Why? Because they are inverse relationships. If you want the very best response time - get a single terminal S/64. Whenever the user requests a task, all of the S/64's resources are at its disposal. The system throughput and price/performance suffers however, since the machine remains idle most of the time. Response time suffers when you try to optimize system throughput. I am always reminded of the image of lines at the supermarket. The maximum system throughput occurs when there are too few cashiers, and lines start to form. The cashiers are fully utilized (ie. always busy) but response time (the time to check out and pay) suffers greatly. In the supermarket though, you have the choice - to leave your basket and go to the nearest 7 - 11 and forget it. In a computer system you're not quite that fortunate (the break key does have some merit though). Our goal then is to find the right balance between acceptable response times and good system throughput.

I will discuss response time from the interactive point of view since online users are much more sensitive to delays in response. The batch user can merely substitute elapsed time for response time throughout this discussion. Response time is basically the time when a user requests a task, to the completion of that task.[2] The factors that determine that time are; data communications rate, system time, and terminal rates. I will leave datacomm and terminals for other discussions and will focus on the system component. This component can be split into 4 pieces; CPU, Memory, I/O, and the application. I will focus only on the first three, since the effects of applications on performance could fill many volumes, as we would need to delve into IMAGE, KSAM, COBOL, WORD, RAPID, graphics, Manufacturing software, Financial software ... [3]

The components we will discuss are important because the interaction of these resources control the length of response time. To improve response time, reduce this interaction time! In order to do this we must first understand how these resources effect response time, how their utilization can be measured, and finally how their utilization can be reduced. Throughout this discussion, percentages and guidelines will be indicated where possible. These are by no means the only "accepted" ones, but rather represent the opinion of the author.

CPU

The central processing unit (CPU) is the controller of the system. It processes tasks on a priority basis and can work on only one task at a time. The objective of effective resource utilization can best be met by utilizing the CPU and I/O resources concurrently. The profile of most transactions is depicted in figure 1.

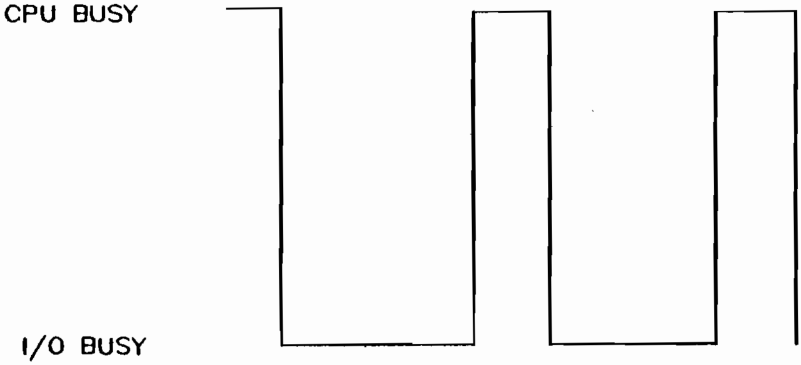


Figure 1

When the cpu schedules I/O for a process, it then becomes available for the next process which is "ready to run". If there are none, the cpu waits - until a process becomes ready, or the process executing I/O completes and it becomes ready to run. For a single user system, this would be like boiling a pot of water for tea, and standing there waiting for that event to occur - terribly wasteful. On the other hand, when many processes require the cpu simultaneously, most will have to wait for their time slice. Can you remember going to the bank, Friday the 31st of the month at 12:00 noon? Even though your transaction will only take 30 seconds, you wait in the line (queue) for your turn.

(NOTE: MPE does give you some flexibility by use of the scheduling queues to determine which processes get preferential treatment. For example, most users setup C Limit to have better priority than D Base Therefore any interactive request is satisfied before all batch requests. The only time a batch process will run is when there is no interactive process ready to run). We measure the effectiveness of cpu utilization by the percent of time a process makes a request, and finds the cpu busy with some other task (CPU busy %). This can be measured by OPT/3000 (On-line Performance Tool). A value of 80% or greater represents fairly heavy utilization.

HOW TO OPTIMIZE CPU UTILIZATION

In situations like these, the solution is to reduce contention for the CPU. This can be accomplished a number of different ways. Load management techniques can be implemented that would schedule heavy loads to non-peak timeframes (overnight, weekends). Another often used approach is to make database updates to a file that gets streamed during non-peak hours. This has the advantage of allowing maximum accessability to the database during high activity periods. The disadvantage is the database is not always in the most current state. Applicaton programs could be evaluated for redundant or sub-optimal code. This is an area that sometimes raises the neck hairs of programming staffs. You want to examine MY code? In many cases, tight scheduling has forced the elimination of the optimizing phase of the project. Since many applications proliferate to installations throughout the world, I am of the firm belief that optimizing that code once, will save an inordinate amount of execution time at end user sites. When you multiply that by the number of times the application will be run, the cumulative effect is staggering. (NOTE: This whole issue has received a lot of attention with the advent of programmer productivity aids. Typically these aids dramatically reduce the time needed to produce code. However, they also produce code that is not as efficient as an experienced programmer could.) APS/3000 (Application Program Sampler) can be used to isolate what segments (or even lines of code within segments) when optimized will leverage your efforts. Achieving 100% improvement in a segment that executes 1% of the time is not as good as a 1% improvement that executes 100% of the time! After these alternatives have been evaluated, the analysis sometimes indicates it would be more cost effective just to get a faster cpu to reduce the utilization.

This is not the case however if the utilization is under 30%. Additional horsepower will buy very little improvement. Referring back to figure 1, if the cpu component is 50 milliseconds, and the I/O component is 400 milliseconds (10 I/O's at 40 MSEC per I/O), what improvement can be expected if we double the cpu power - 25 milliseconds. By and large, this effect is negligible. As the old saying goes "all cpu's wait at the same speed!" Additional horsepower may complete the cpu component in half the time, yet while the process waits for other resources, the cpu also waits!

MEMORY MANAGEMENT

The effects of memory management on performance have been well documented so I won't spend much time here except for a few observations. With the price of memory dropping it does not make much sense to be underconfigured (some users with S/30/33's and S/II's are somewhat constrained). In fact, memory really only comes into play as a major component when it is underconfigured. In order to execute, a process requires that its stack, a code segment, and any extra data segments that it needs immediately be in memory. When a process references a segment that is not in memory (an absence trap), that segment must be brought into memory before the process can continue. If there is enough free memory, the memory manager allocates a free region for the segment and brings the segment in. If there is insufficient memory, a segment must be removed from memory (swapped out) or overlaid (only code segments or unmodified data segments can be overlaid without being written out to disc). This whole procedure is called swapping, and is controlled by the memory manager. When this situation is particularly aggravated, we call it thrashing. Thrashing could be defined as the memory manager busily swapping segments, but no useful work getting done. When a process is waiting for a swap to occur, and the cpu has no other process ready to run, we say the cpu is "paused for swap". A value of 10% or less is considered normal, 10 - 15% cautionary, and greater than 15% is serious. I'd like to make an interesting point on how numbers can be deceiving. I can guarantee a system would never show these percentages for paused for swap by running an idler program in the EQ that grinds cpu (perhaps a basic program: 10 GO TO 10). When memory pressure occurs and a process is waiting for a segment to be brought in (pause for swap would normally occur), the cpu would have a process ready to run and would therefore show this time as cpu busy time! I make this point because it does not have to be the idler that causes this, any heavy cpu utilization will demonstrate the same trend. Another indicator of memory pressure is the percent of disc I/O generated by the memory manager. When 25% or more of all I/O is generated by the memory manager, memory shortages are occuring. Much has been written about techniques to utilize less memory for those installations that are maxed out. These techniques range from shrinking the stack where possible, to removing static tables from the stack to code segments.

DISC I/O AS A BOTTLENECK

For today's systems the most important performance factor is clearly disc I/O. Discs are rotating mechanical devices and therefore operate at much slower speeds than cpus or memory. It is no wonder that disc I/O easily becomes a bottleneck. There are two ways to reduce this contention: complete each I/O faster, and request fewer I/O's!

Every disc I/O contains the following components;

	Set up Overhead	Seek	Latency	Transfer	Total Avg Access
7920	1.1	25.0 MS	8.3 MS	737 KB/sec	34.4
7925	1.1	25.0 MS	11.1 MS	737 KB/sec	37.2
7933 no PEP	10.3	24.0 MS	11.1 MS	1060 KB/sec	45.4
7933 PEP	3.8	24.0 MS	11.1 MS	1060 KB/sec	39.0

NOTE: PEP represents an enhancement to the CS-80 firmware which reduces the execution time to issue commands to CS-80 drives. Effectively, every CS-80 disc access with PEP completes significantly faster than its non-PEP counterpart.

All discs have these components, and these speeds are the technological speeds available for discs today. When disc I/O requests are processed serially, the time to complete each I/O will be the average access times listed above. If multiple users request I/O, the sequence of events is depicted in figure 2. We will use 40 MS as the average disc access time.

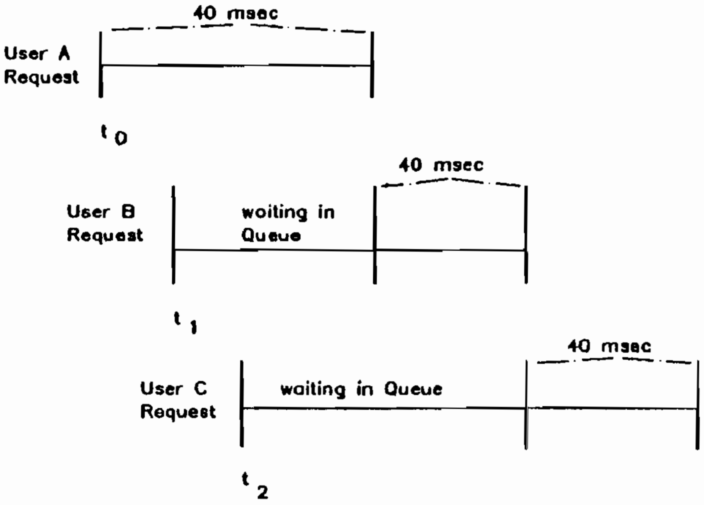

40 msec
User A Request
t_0
40 msec
User B Request
waiting in Queue
t_1
40 msec
User C Request
waiting in Queue
t_2

Figure 2

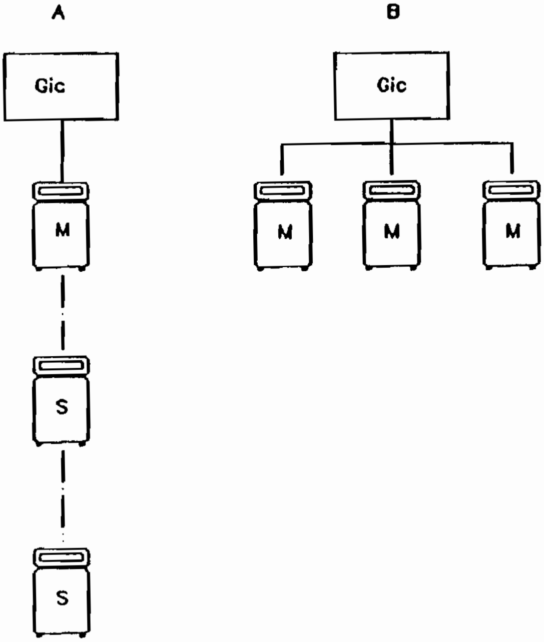
Notice that any user that makes a request while another user's request is being processed, waits until that request is completed before it is scheduled. The result then is the cumulative time! With MPE IV, changes were made to the I/O system that enable dramatic improvements in this area.

LOOK-AHEAD SEEKS

The S/III has a feature called look-ahead seeks. When enabled, this allows the user to have multiple seeks scheduled simultaneously, so up to 4 discs can be seeking at the same time. When the disc controller is being scheduled to seek, it checks the disc request queues for up to 4 discs to see if they can be scheduled at the same time. Once the seeks are completed, the transfer still remains serialized, but we've eliminated serialization of the longest component (the seek).

OVERLAPPED SEEKS

This same philosophy can be accomplished on the HPIB systems (S/30/33/39/40/44/64) by using master disc drives (a master disc has a disc controller attached to it). This technique is called overlapped seeks.



The above diagram illustrates the difference between a master-slave configuration, and an all master disc drive configuration. System A's performance would be similar to that depicted in figure 2. While the controller is waiting for the I/O request to complete, all other requests are queued. System B however can schedule each disc drive independently since each has a controller. There is still the potential for a bottleneck at the GIC level, since only one controller can transfer at a time, however testing has shown with two masters/GIC negligible contention occurs, with slow rising contention from 3 to 4 masters/GIC, at which point the channel basically becomes saturated (see figure 3). That does not mean you cannot attach 5 or 6 masters/GIC, merely that there is less benefit of devices 5 and 6 being masters - they could be slaves and would perform close to as well. Figure 3 illustrates average I/O rates for the 7925 disc drive.

CONFIGURATION	I/O'S / SEC	
1 M, 1 G, 1 I	22	
1 M, 2 G, 1 I	22	
2 M, 1 G, 1 I	44	
2 M, 2 G, 1 I	44	
3 M, 1 G, 1 I	53	
3 M, 2 G, 1 I	66	
4 M, 1 G, 1 I	60	
4 M, 2 G, 1 I	88	
5 M, 2 G, 1 I	94	
6 M, 2 G, 1 I	104	
4 M, 2 G, 2 I	88	ALL 2 IMB DATA IS
4 M, 4 G, 2 I	88	ESTIMATED! THE
5 M, 3 G, 2 I	110	SALES CENTER HAS
6 M, 3 G, 2 I	132	NOT TESTED THIS!

LEGEND:

M = 7925 MASTER DISC DRIVE (10 - 30% less for the 7933 without PEP)
G = GENERAL I/O CHANNEL (GIC)
I = INTER-MODULE BUS (IMB)

Figure 3

HOW CAN THE USER CAPITALIZE ON THIS?

The best way to capitalize on these features is to spread requests across discs to gain concurrency of seeking, as well as a concurrency of I/O with CPU. In a single user system (batch job), no overlap can occur. If all users wish to access the same file, or a set of files that exist on the same disc - NO OVERLAP CAN OCCUR! How can a user determine this though? The best way I've been able to find is a set of contributed programs called "DISCIO". These programs analyze the log files looking for type 5 (File Close). They will format the file open/close activity based upon number of times this occured, and will also indicate the disc drive the file resides on. This will usually give a good indication of which files are most heavily utilized.

A common rule of thumb is 80/20. 80% of all accesses are to 20% of the files (some people say with respect to disc files this is more like 93/7!). OPT/3000 will indicate which drives are most heavily utilized, and therefore that imbalances exist, but not which files. This philosophy of spreading accesses across discs pertains particularly well to virtual memory, as it flattens the disc drive distribution. (This used to be a big problem with heavy contention for LDEV 1 prior to MPE IV).

Another Strategy that seems to be effective is to split heavily used master and detail IMAGE datasets across different discs. If they are on the same disc, you are guaranteed a certain amount of head movement will occur to access a record. However, if they are split, there's a chance that subsequent accessess will not have to reposition the head each time. The amount of benefit this buys you depends on the magnitude and type of accessing being done. Finally, some have suggested placing your heavily used files toward the middle of the disc. The philosophy is, on average you will traverse one third of the disc per access. If you place these heavily used files in the middle of the disc, generally you can shorten this seek distance. In theory this sounds plausible, in reality it is not easily accomplished. MPE allows you to choose what disc drive, but not what cylinder or track. The previous suggestions do however work well, and please note: once you find an optimal placement strategy, back it up! Remember when reloading to use the RESTORE option, or you efforts will have been wasted. (For double protection map out your file placement strategy and keep it in a safe, accessible location!)

DISC I/O CACHING

When I first heard the disc I/O optimizing principle - request fewer I/O's I can remember chuckling, how absurd! If I could get by with fewer I/O's don't you think I would? That's the whole issue of adding memory isn't it, so there would be fewer I/O's necessitated by the memory manager? Yes, in actuality it is. Now we see much greater strides in thinking we call "Disc I/O Caching". The principle behind caching is to eliminate PHYSICAL I/O's and replace them by LOGICAL I/O's. Whenever an I/O request is made, it takes approximately 40 MSEC to complete. Of that time, only 1 MSEC/1000 bytes is for transferring data. By utilizing excess main memory as a cache for data with a high probability of access we can eliminate physical I/O's. When a physical access is made to the disc, instead of only transferring the requested data size, an additional amount is also transferred to the cache for future use. Clearly this will be very effective for serial access, however we have found even with random access locality does exist (the 80/20 rule again). What is the cost of this? The additional overhead necessary to transfer additional data (for 16KB this is 16 MSEC). The system can make a lot of mistakes (transfer data that turns out to be unneccesary) as long as it makes a couple of hits (the average cache access time is less than 5 MSEC/access). Preliminary testing has indicated greater than a 75% hit ratio! In order for this to work however, there must be adequate memory to utilize for this purpose. The cache regions are treated like any other memory region and are subject to being overlaid.

If memory pressure exists, the hit ratio is likely to drop. Initial estimates indicate about 1 MB of main memory is necessary for this technique to be effective. To date, not enough testing has been completed to many any definitive statements.

HOW DOES DISC I/O CACHING AND MASTER DISC DRIVES JIVE?

The combination of disc I/O caching and master disc drives makes an interesting question. In other words, is it better to totally spread related data across master disc drives, or to bunch them on the same drive and capitalize on caching? Intuitively, my guess would be data that will be accessed simultaneously (or close to it) should be bunched together to allow the caching to be effective, since the access time savings is substantial. This could be done by overtly specifying the drive and using naming conventions like AA1, AA2, ... Since everything can't be bunched together, other related items should be spread across disc drives. This way, when physical accesses are required to replenish the cache, they will be completed expediently. In this way we accomplish both goals, complete each I/O faster, and request fewer I/O's!

One last subject should be addressed, that of resource contention for file locks, RINs, and other system tables. Both OPT/3000 and APS/3000 have the capability to isolate many of these deadlocks (TUNER2 for key system tables as well) and should be utilized to determine the causes of bottlenecks. A general rule is: hold resources for the minimum time necessary. This merely requires thinking through the way an application will use these resources from a global perspective.

In conclusion, techniques are available to measure the main system resources; CPU, memory, and I/O. Once the resource utilization is calibrated, bottlenecks can be isolated and steps take to alleviate contention. Through overlapping resource utilization, both system level goals can be accomplished; increasing system throughput and therefore reducing user response times. There is a balance between which of these requirements has higher priority, and that will determine the exact techniques utilized to accomplish that goal. In many cases, implementation of the above mentioned techniques can eliminate (or at least delay) the need for additional equipment in order to meet increasing end user demands.

HEWLETT PACKARD

BUSINESS DEVELOPMENT GROUP 19447 Pruneridge Avenue, Cupertino, California 95014 Telephone 408 725-8111

BIBLIOGRAPHY AND FOOTNOTES

1. Colloquial for "buy more hardware"

2. For a more detailed description of response times see "Benchmark Techniques for Characterizing Application Performance", David S. Wertheim (also presented at this conference).

3. A partial bibliography is listed below

Davison, Gerald, "Image Locking and Application Design", Journal of the HPGSUG.

Hulme, John, "System Design and Optimization Techniques and Tools", Journal HPGSUG, Third Quarter 1980, Vol 3, No. 3.

Rego, Alfredo, "Database Therapy: A Practioner's Experience".

White, Fred, "Improving Performance of IMAGE applications", Journal HPGSUG, May 1979, Vol 2, No. 4.

4. Greene, Robert, "MPE internals for Neophytes", Interact, July/August 1982, Vol 2, issue 4.

5. Greene, Robert, "Optimizing On-line Programs", Technical Report, June 1981.

6. Howard, Jack, "System Design and Optimization Techniques and Tools", Journal HPGSUG Third Quarter 1980, Vol 3, No. 3.

7. May, Jim, "Programming for Performance", Journal HPIUG, July/December 1982, Vol 5, No. 3,4.

8. Squires, Jim and Splinter, Ed, "System Performance Measurement and Optimization".

9. APS/3000 (Application Program Sampler) Reference Manual and User Guide, June 1982 (Part No.31280-90001)

10. OPT/3000 (On-line Performance Tool) Reference Manual, August 1981, Part No. 32238-90001.

Architectural Changes for MPE V

David N. Holinstat
Development Engineer
Hewlett-Packard GmbH
Böblingen, Baden-Württemberg
Federal Republic of Germany

ABSTRACT

MPE V supports 255 code segments per program, as well as allowing most MPE Tables to reside anywhere in the first 4Mb of memory. These expansions necessitated several modifications to the HP3000 architecture. Relocation of MPE's Tables required changes to the LST and SST instructions. Removal of the code segment limitations and expansion of the Code Segment Table involved adding a new table, the Logical Segment Transfer Table (LSTT), to the HP3000 architecture. In addition, alterations were made to the PCAL, SCAL, EXIT, IXIT, PARC, ENDP, and XBR instructions, the Segment Transfer Table (STT), and one bit in the Stack Marker.

The presentation will quickly review the original HP3000 architecture, then explain the alterations for MPE V.

ARCHITECTURAL CHANGES FOR MPE V

David N. Holinstat

Since its first introduction, the HP3000 architecture has changed several times. For example, the Series I and its predecessors allowed a total of 255 code segments for the entire machine, including MPE and all users. When the Series II was introduced, the architecture was modified to provide 192 code segments for MPE, plus up to 63 code segments for every program. Series II also provided 64-bit Extended Precision Floating Point, an improvement over the 48-bit precision found on Series I. Introduction of the Series 33 brought an entirely new I/O system to support the HP-IB peripherals.

The most recent change, part of the recently announced Series 42, 48, and 68 systems, will remove many of the current "maximums" caused by the HP3000 architecture itself. These architectural changes, along with the new MPE V operating system, will allow each program to have 255 code segments (private or shared), as well as allowing many critical MPE tables to double or quadruple in size. This will allow many more jobs, sessions, terminals, programs, etc., to be supported as were on MPE IV.

General Table Expansion

"MPE has a table for everything." The MPE Tables manual currently has 20 chapters and is several hundred pages in length. All the data required to control user processes, check security, handle memory and I/O resources, etc., is kept by MPE in its tables. So, as the number of terminals, sessions and jobs supported has grown, HP has needed to expand MPE's tables. However, for reasons which will become apparent, many of MPE's tables must be kept in Bank 0 of memory. And recently, as the number of terminals and sessions allowed quickly grew, it became apparent that Bank 0 is not an unlimited resource!

Historically, MPE has kept its data in an area known as the System Global Area (SYSGLOB), beginning at location %1000 in Bank 0 of memory. When MPE wanted to access this data, it would set DB to 0.%1000. (We write "0.%1000" for Bank 0, loc. %1000.) It could then access tables (which are just arrays that happen to contain MPE's data) using the normal LOAD and STOR instructions (Fig. 1a). These instructions can directly address DB+0 through DB+255, so MPE keeps pointers to the most important tables and a few very commonly used variables in this area (from %1000 to %1377). When DB is set to SYSGLOB, many tables can be accessed with normal LOAD and STOR Indirect instructions. For example, suppose location %1005 points to the I/O Request Queue (IOQ). We can access the 20th word in the IOQ with the following code sequence:

```
    Instruction           Function
(a) LDXI  20              X := 20;
    LOAD  DB+5,I,X        TOS := ((DB+5) + DB + X)
                              = (%4000 + %1000 + %24)
                              = (%5024);
```

[Please see Fig. 2 for the sample SYSGLOB layout to be used in these examples. Note that "(%nnnn)" means the contents of loc. %nnnn. Also, since all Store-type instructions function exactly like the Load-type instructions as far as address calculation goes, all examples use the Load-type instructions.]

It became apparent that it was very uneconomical to always switch the DB register of SYSGLOB when accessing system tables, because the actual act of switching can take significant CPU time. Thus, the Load System Table (LST) and Store System Table (SST) instructions were designed. These instructions reference tables in SYSGLOB without actually switching the DB register (Fig. 1b). So, code sequence (a) can be replaced by sequence (b), without regard to the current content of the DB register:

```
(b) LDXI  20              X := 20;
    LST   5               TOS := ((SYSDB+5) + SYSDB + X)
                                = (%4000 + %1000 + %24)
                                = (%5024);
```

The addressing range of LST is 0-15, because only 4 bits could be spared in the opcode. (Such are the problems with adding instructions after the original design!) So, in order to make the instruction more useful, LST 0 was given a special meaning. When an LST 0 is executed, the offset to the array pointer is taken from TOS, thus giving an even greater range than the other Load instructions. For example, reading the 30th word of the Example Table (pointer at DB+134) could be done as follows:

```
(c) LDXI  30              X := 30;
    LDI   134             TOS := 134;
    LST   0               TOS := ((SYSDB+(TOS)) + SYSDB + X)
                                = ((%206) + %1000 + %36)
                                = (%101400 + %1000 + %36)
                                = (%102436)
```

Examples (b) and (c) certainly require as many or more instructions as does (a). However, they don't require DB to be set, an operation taking dozens (possibly hundreds) of instructions.

There is one fault, however, with both of the above approaches to system table access. Whether we use the normal LOAD instruction with DB-relative addressing (implying DB set to 0.%1000), or the LST instruction, which addresses relative to SYSGLOB (0.%1000) implicitly, we have one critical limitation: we are always addressing somewhere in Bank 0. This means that any tables addressed using one of the preceding methods must be in Bank 0. When you recall that a bank of memory on the HP3000 contains exactly 65,536 words, you realize that as MPE grows larger, we will have a problem finding space for all of the tables. And the need for space becomes particularly acute when adding terminals and sessions, since each terminal requires one entry in each of several I/O-related tables, and each session requires entries in many job-related tables.

One possible solution would be to use absolute addressing for all tables. There are problems with this, however: First of all, our degrees in Computer Science would all be retroactively revoked! More seriously, an absolute address takes 2 words, one for Bank and one for Offset. This means that the 255 locations in SYSGLOB that are directly addressable could hold only 127 pointers maximum, instead of the 255 possible today. Also, some tables contain cross-references to several other tables. Today these cross-reference pointers are usually SYSGLOB-relative. Thus, a table entry containing pointers to 5 other tables would have to be 5 words longer to hold absolute address pointers to the 5 other tables. If the table has 100 entries, we have just used 500 more words! So, absolute addressing was ruled out to the extent possible.

Instead, it was decided to change the format of system table pointers, and change the LST and SST instructions as well. Instead of a 16-bit SYSGLOB-relative pointer, LST and SST will now expect a pointer with the following format:

```
| Base.(0:11)  | Bank    |
|______________|_________|
0            10 11     15
```

Bits (11:5) specify the bank in which the table will be found. The base address is calculated by setting the bank bits (11:5) to zero and adding %1000. Note that the base address is calculated as it currently is, except that the low order 5 bits are set to zero. The new format has several ramifications:

1) The use of 5 bank bits allows MPE tables to be loaded anywhere in the first 32 banks (4 Mb) of memory.

2) MPE tables are required to begin on 32-word boundaries, since the last 5 bits of a table address will always be zero.

3) Compatibility with Series II, III, 30, 33, 40, and 44 is maintained. Separate versions of MPE will not be required.

Point 3 needs some clarification. Beginning with MPE V, INITIAL will always create those tables referenced via LST/SST on 32-word boundaries. However, it will also determine whether it is running on a machine with MPE V level firmware or not. If not, INITIAL will simply locate all such tables in Bank 0. The old LST instruction then sees a 16-bit pointer whose low order bits happen to be 0; this points (correctly) to the beginning of the table. If one were now to install the MPE V firmware and do a COOLSTART, the new firmware will see the same table addresses, this time pointing specifically to Bank 0. So, MPE V will run on both old and new firmware without problem. Of course, if MPE V is configured with larger table sizes than can possibly fit in Bank 0, and is so installed on a machine with the old firmware, you will get the old, friendly "OUT OF MEMORY" message from INITIAL!

Code Segment Table Expansion

Recently some HP3000 users have attempted to run several large applications systems on the HP3000 simultaneously -- and have been rewarded with the message "OUT OF CST ENTRIES--UNABLE TO LOAD PROGRAM TO BE RUN". Upon examination of the configuration, they find that the Code Segment Table is configured to its maximum 192 entries. These users have often asked, "Why doesn't HP just change the maximum size of this silly table?" Unfortunately, expanding the CST is not just a matter of changing the size of an array in 2 or 3 MPE modules and recompiling! It has taken a major revision of MPE and of the 3000 architecture to allow each program 255 code segments. The following section will explain the Code Segment Table (CST) structure, both new and old, as well as the difficulties involved in changing it.

Let's quickly review the concept of a code segment. All memory on the HP3000 is divided into code segments, data segments, and free areas. All program code is kept in code segments, and all data in data segments. But these segments can be located anywhere in memory. So, to begin running a program in code segment 2, we first have to find code segment 2. To do that, we look in the Code Segment Table (CST), where there is a four-word entry for code segment 2 (Fig. 3). Among other things, this entry tells whether the segment is in memory ("Present") or on disc ("Absent"). If present, the CST entry tells where the code segment is located and how long it is. To begin executing code within this segment (after a PCAL or EXIT), the 3000 sets PBank := Bank, PB := Base, P := Base + desired offset into segment, and PL := Base + (Length/4)*4. The format of the Code Segment Table entry hasn't changed much since the original 3000. What has changed is the way in which we find the appropriate entry. In other words, the layout of the CST has changed, but the content remains the same. Changing the layout once more with the release of the new Series 42-48-68 machines has enabled us to remove the 192 entry limitation, as described below.

The first point to discuss is a very important limitation: A code segment is identified by its number, which is between 1 and 255. This limitation comes from the fact that only 8 bits are allowed for the segment number (Seg#) in the Status register. Changing this limitation directly would cause incredible problems! For example, suppose we wanted to quadruple the addressing range to 1024, which would require 10 bits for Seg# instead of 8. First, to hold the extra bits, we would have to make the Status register an 18-bit register. (Those of you with microprocessor experience know how easy it is to find a register chip with 18 bits!) Furthermore, the extra 2 bits would have to go in the stack marker. We can get one bit reasonably easily, but not two; therefore, we would have to go to a 5-word stack marker. Changing to a 5-word stack marker would mean that all of MPE and all user programs in the world (!) would have to be recompiled. Except, of course, SPL programs, which would have to be rewritten! Given that HP likes to brag about compatibility, this "brute force" method of adding more code segments can be quickly rejected.

The Code Segment Table architecture has been modified once before. The Series I and its predecessors had the simplest CST structure. At any given time there were 255 code segments available on the machine, numbered 1 to 255 (Fig. 4). The total number of code segments assigned to MPE and all running programs could never exceed 255. Of course, since the 3000 was introduced with a maximum memory size of 64K words, this was quite reasonable. As the system grew and more users were added, resources became more precious. One result was the first CST expansion, which was introduced with the Series II.

With the advent of the Series II, the Code Segment Table was divided into 2 domains: the system Code Segment Table (CST) with entries for segments 1-191, followed by the Code Segment Table Extension (CSTX) (Fig. 5). In the CSTX are blocks of entries, one block per program. Each block contains entries for segments 193-255 (%301-%377). MPE segments are assigned entries in the system CST, as are user segments that come from a Segmented Library (SL). Program segments have their entries in the CSTX. So, if a program has 6 code segments, there will be a block of entries in the CSTX for segments %301-%306. A program having 63 segments will have a CSTX entry block containing entries for segments %301-%377. A pointer in absolute memory loc. 1 points to the block of CSTX entries in use at any given time; this pointer is updated by the MPE Dispatcher whenever it launches a process. This architecture is the basis for the current CST limitations: 191 system and SL code segments, and 63 code segments per program. One can easily imagine 20 programs running, each with 63 code segments; this is a vast improvement from the 255 code segment maximum on Series I.

However, there is still a problem with the above method. As noted, there is a block of CSTX entries for each program. But only code segments that are part of a program can have entries in the CSTX, because these segments are only accessible when that program is running. Sharable segments--i.e., segments coming from an SL--are not associated with any particular program. Therefore, to allow them to be accessed from any program, they must be given entries in the system CST. Since MPE can use up to 100 of the 191 available entries, there is clearly a rather strict limitation on the number of SL segments a program can use. Unfortunately, large application systems, such as HP's own MM/3000 and HPFA, use many sharable segments in the interest of efficiency: If a code segment is included in 5 different programs (for example, using an RL), then when all 5 programs are running, 5 copies of the segment must reside in memory; but if the segment is put into an SL, it is made sharable, and one copy can be used for all 5 programs. This takes, however, another CST entry. So, for MPE V, a scheme was developed to allow more entries in the CST by introducing the concept of "code segment mapping".

Under the new system, a code segment can be "logically mapped" or "physically mapped". Physically mapped code segments will be located as they are today--with entries from 0 through 255 beginning at the CST Base address stored in absolute loc. 0. There can only be 255 physically mapped code segments, and these will all be reserved for MPE. All user segments will be logically mapped, including subsystems such as EDITOR and COBOL. Program segments will have entries in the CSTX, similar to the current (Series II -> 64) systems. Sharable user segments (i.e., SL segments) will have entries in the CST with physical segment numbers of 256 and greater. A CPU-internal flag will control the current mapping state of the CPU: physical or logical. This will not be held in the Status register, because (as mentioned previously) there is no space available there.

When the CPU transfers control to a logically mapped code segment, it looks first at a new table, the Logical Segment Transform Table (LSTT) (Fig. 6). Each program has its own LSTT, which is pointed to by absolute loc. %1221-%1222 whenver that program is running. (The Dispatcher updates these locations, as well as loc. %1223, the number of CSTX segments in the program, whenever a process is launched.) To transfer to logical segment 2, the CPU looks at Entry 2 in the current LSTT, which contains the physical segment number of the target segment (Fig. 7). The CPU then uses this physical CST number to index into the system CST, where the appropriate entry points to the code segment in the usual manner. This physical segment number can be as high as 2047, which is the new maximum number of CST entries.

When the CPU transfers control to a new code segment, it must determine whether the segment is physically or logically mapped. To do this, changes were made to some familiar architectural structures: the STT (Segment Transfer Table) and the Stack Marker.

To explain the differences between the old and new approaches, we should first look at a few examples of the old (current) method. PCAL will be used as an example; refer to Fig. 8 for illustration.

Suppose a PCAL 1 is executed. First, a Stack Marker is written on the stack. Then, the CPU checks to see if 1 (the PCAL operand) is less than or equal to the number of Plabels in the STT (STT Length), found at PL-0. If so, the Plabel at PL-1 is read. Bit 0=0 signifies that this is a local Plabel, in the format shown. If Bit 1 (U)=1 the segment is uncallable, and user mode callers will abort with an STT Uncallable Violation. Finally, Bits 2-14 are a PB-relative offset into the current segment. After bounds checking assures that (PB <= PB + new Delta-P <= PL), PB and P are set to the new values, which transfers control to the new procedure.

Now, suppose a PCAL 5 is executed (Fig. 9). A stack marker is written, and if 5 <= # of Plabels, PL-5 is read. Bit 0=1 signifies that this is an external Plabel. That means the procedure we want is in another code segment, and the first order of business is to find that other segment. The segment number is in the Plabel, bits 8-15. (Note that Seg# is only 8 bits long here as well!) If Seg# is less than 192, the CPU looks in the CST; otherwise it looks in the CSTX. The 4-word CST entry pictured in Fig. 3 begins at the following address:

For CST: Seg# * 4 + CST Base
For CSTX: (Seg# - 192) * 4 + CSTX Base

The CST Entry tells the CPU if the segment is present or absent. If absent, the PCAL stops here, and MPE is awakened via an Absence Trap. If the target segment is present, the CST Entry gives the absolute starting address (Bank and Base) as well as the segment length. Assuming the segment is in memory, the new PL value is calculated, and the Plabel at (new PL - STT#) is read, where STT# is bits 1-7 of the Plabel already read from the current segment. The new Plabel read from the target segment must be in internal format; otherwise the process will abort with an STT Violation. At this point, bounds checking is performed and control transferred as described above.

With MPE V and the new firmware, the STT will look a bit different (Fig. 10). Bit 0 of a Plabel will no longer be used to signify internal/external, but instead to indicate whether the target code segment is physically or logically mapped. The internal/external determination will be made via word 0 of the STT, which will hold "number of local Plabels" as well as "number of Plabels". The local Plabels are always written at the beginning of the STT, so for a PCAL n the determination is made as follows:

$$n = \begin{cases} 0 \text{ -> Use Plabel from TOS; defined to be in external format.} \\ 0 < n <= \#\text{local Plabels - - -> Local format} \\ \#\text{local Plabels} < n <= \#\text{Plabels -> External format} \\ \#\text{Plabels} < n \text{ - - - - - -> STT Violation} \end{cases}$$

When an internal PCAL is performed (calling a procedure in the current segment), PCAL will operate exactly as today, as described previously. But if an external PCAL is done, the CPU will behave quite differently (Fig. 11a). First, the stack marker will be written and the Plabel at PL-n read as before. If bit 0 of the Plabel is 1, signifying a physically mapped segment, the segment number is used to directly access the CST as before, although the allowable range of segment numbers will now be 1-255 instead of 1-191. The CPU's mapping flag is set, and control is transferred. However, if bit 0 of the Plabel is 0, then the CPU must first check memory loc. %1223, which tells how many segments the program file has. If segment number <= # program segments, then the code segment entry will be found in the CSTX entry block for this program (Fig. 11b). (As previously described, memory loc. 1 points to the current CSTX block.) If Seg# > # program segments, then this is a shared (SL) segment (Fig. 11c); furthermore, the Seg# that we have is a logical number. We must look in the LSTT at loc. (2 * logical Seg#) to find the physical Seg#. This physical Seg# is then used to access the CST, which now has 2047 as largest possible segment number. Note that the Status register reflects the logical segment number, but the mapping flag says "logical mapping". This enables the CPU to know where to look for a given code segment.

Suppose the CPU has been executing in logical segment 5, and then does a PCAL to logical segment 6, where both segments are SL segments with entries in the CST. The Mapping flag shows logical mapping, but this is an internal flag, not accessible to the program. To leave seg. 6, the procedure does an EXIT instruction. EXIT finds the return segment number in the old Status register, stored at Q-1 in the stack marker. However, the old Status register merely says "5" for Seg #; this could be an EXIT to either physical seg. 5 or logical seg. 5. How does the CPU know which? With MPE V and the new firmware, a bit in the stack marker has been reassigned to save the mapping flag (Fig. 12). Bit 0 of the Delta-P (Q-2) has always meant that a Control-Y interrupt was pending; this is set by MPE when a Control-Y interrupt is received. Bit 1 signified that a Trace interrupt was pending, again set by MPE. With the new system, Bit 0 will indicate that either a Control-Y interrupt or a Trace interrupt is pending. Because of the way this was defined in the past, it will be easy for MPE to differentiate between the two. Bit 1 of Delta-P will be the old mapping flag. Thus, the stack marker contains both the segment number and the mapping flag. When the EXIT is executed, the CPU can find the appropriate code segment by getting its entry from the CST or CSTX, using the LSTT if returning to a logical segment. The method used is the same as for PCAL, except that the STT is not referenced. There is no need to find a Plabel, because we already know the target segment number and PB-relative return address.

Some of you who see memory dumps from time to time will notice that the Delta-P values of MPE segments (in an MPE V dump) seem unreasonably large. Actually, you are seeing the old mapping flag laid down in the stack marker. MPE segments are usually physically mapped, and Delta-P Bit 1=1 for physically mapped segments--so it looks as if all MPE segments have a Delta-P of %40000 or larger.

You may have noted in Fig. 6 that the LSTT is divided into 2 parts. In the first part is a two-word entry for each logical segment used in this program, containing the physical segment number and a pointer to an "External Label List". The second part contains an External Label List for those shared (SL) segments which are referenced by this program. The reason is, the STT of a sharable segment cannot contain a logical segment number in an external Plabel since more than one program may be sharing that segment.

Computer Museum

For example, imagine the following procedure in a sharable segment:

```
procedure A;
begin
  B;
end;
```

Imagine that B is also in a sharable (SL) segment. Now, I run PROGX, which has 2 segments and calls A. The loader will set up the STT in PROGX such that segments 1 and 2 are the PROGX segments, Seg. 3 contains A, and Seg. 4 contains B. Now, since A calls B, the STT of A must reflect the fact that B is in Seg. 4. (See Fig. 13) So far, so good---but now you run PROGY, which also calls A, and has only one segment. The loader must assign the number "1" to the PROGY segment, and "2" and "3" to the segments containing A and B. But now, the STT for A must reflect that B is in logical Seg. 3---which is somewhat difficult, since it must also show that B is in logical Seg. 4!

The problem is solved by using "0" as the logical segment number of every sharable segment called by another sharable segment. In other words, the STT of a sharable segment contains 0 as the Seg# for those Plabels referencing other sharable segments. When the CPU encounters "logical seg. 0" during a PCAL, it looks at the LSTT to find the External Label List for the current segment.

Using the above example, when A does a PCAL n to get to B, the CPU finds a 0 for Seg# at PL-n. It then looks in the Status register to get the current logical Seg#, and reads LSTT(2*Seg#+1). It can then index into the External Label List, which contains all the External Labels for this particular execution of the current segment; there it will find the correct Plabel. [The Plabel is actually found at (beginning of External Label List) - n + # local Plabels. This is because only the external Plabels are there, so we add in the number of local Plabels to effectively "skip over" the nonexistent local ones.]

In the preceding examples, only PCAL and EXIT have been discussed. Several other instructions were also changed to support the new CST/STT accessing scheme. These changes are briefly described below:

LLBL - Fetches a Plabel in the same manner as PCAL, accessing the LSTT if necessary.

IXIT - Performs a transfer of control just like EXIT (in fact, it executes the same microcode).

SCAL - Works the same as it always has, but has to know about the new STT format to know if the target Plabel is internal or external.

Interrupts - Interrupts do implicit PCAL's to MPE interrupt routines. These implicit PCAL's work just like ordinary PCAL's.

COBOL 74 instructions - The instructions XBR, PARC, and ENDP are special intructions generated by the COBOL 74 compiler to create more efficient COBOL object code. They provide a

> method of transferring directly from one code segment relative address to another, using a segmeng # and offset instead of an STT with a Plabel. Hence, these instructions resemble EXIT in their method of control transfer. When the transfer is external (to another code segment), the same considerations are in effect--the LSTT must be consulted if the target segment is logically mapped.

We've discussed the architectural changes necessary to support the advances in the MPE V operating system. However, it should be noted that the bulk of the engineering work went into improvements to MPE itself, not into the firmware changes described here. It is beyond our scope to discuss all of the improvements to MPE, but the list is substantial, and hundreds of "engineer-months" were required to complete the project. Other presentations will discuss methods used to coordinate all the work done on MPE, as well as some of the implications of the new software.

I couldn't close without thanking Dan Mathias of CSY Software R&D for his help, both in helping me to understand the whole mess (!), and in permitting me to use some of his working documents in this paper.

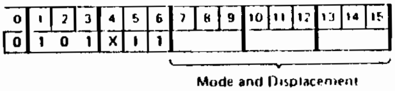
SELECTED HP3000 ARCHITECTURAL DEVELOPMENTS

	Series I	Series II, III	Series 30,33,40,44,64	Series 42, 48, 68
Code Segments (maximum)	255	192 SL (incl. MPE) +63 per program	192 SL (incl. MPE) +63 per program	255 MPE +255 per program (incl. SL)
I/O	Parallel/Differential (SIO)	Parallel/Differential (SIO)	HP-IB	HP-IB
Extended Precision Floating Point representation	48-bit	64-bit	64-bit	64-bit
MPE Memory Resident Tables				
must reside in:	Bank 0	Bank 0	Bank 0	Banks 0 through 31
accessed via:	LOAD/STOR	LST/SST	LST/SST	modified LST/SST

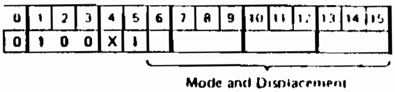
SLIDE I

Ⓐ

STOR Store TOS into memory. The content of the TOS is stored into the effective address memory location, and is then deleted from the stack.
Memory opcode 05, bit 6 = 1
Indicators: unaffected
Addressing modes DB+, Q+, Q−, S− relative
Direct or indirect
Indexing available
Traps STUN, BNDV

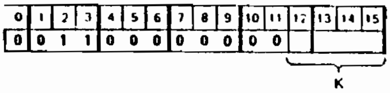

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 0 1 X 1 1
Mode and Displacement

LOAD Load word onto stack. The content of the effective address location is pushed onto the stack.
Memory opcode 04
Indicators CCA
Addressing modes P+, P−, DB+, Q+, Q−, S− relative
Direct or indirect
Indexing available
Traps STOV, BNDV

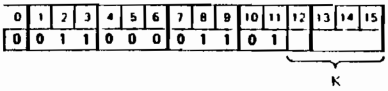

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 1 0 0 X 1
Mode and Displacement

Ⓑ

LST Load from system table. The X register contains a value which is used to index into a table pointed to by the contents of location %1000+K if K is non-zero, or by the contents of location %1000+A if K is zero. The table pointer itself is also relative to location %1000. The data accessed in the table is pushed onto the stack if K is non-zero or replaces A if K is zero.
Special opcode 00
Indicators CCA
Traps STUN, STOV, MODE
This is a privileged instruction.


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 0 1 1 0 0 0 0 0 0 0 0
K

SST Store into system table. The X register contains a value which is used to index into a table pointed to by the contents of location %1000+K if K is non-zero, or by the contents of location %1000+A if K is zero. The table pointer itself is also relative to location %1000. The data contained in A if K is non-zero or in B if K is zero is stored into the calculated address. The stack is then popped by one if K is non-zero or by two if K is zero.
Special opcode 15
Indicators unaffected
Traps STUN, MODE
This is a privileged instruction


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
0 0 1 1 0 0 0 0 1 1 0 1
K

Ⓒ For the new LST and SST, the phrase "The table pointer itself is also relative to location %1000" is replaced by "The table pointer contains the bank of the table in bits 11:5. The base address is computed by taking the 16-bit table pointer, setting bits 11:5 (the bank address) to 0, adding %1000, then adding X."

Figure 1

Referencing the System Global Area (SYSGLOB)

Example Layout

```
Memory
Loc.        ____
     %1000 |
      1001 |
           :
      1005 |  %4000 (SYSDB-relative pointer to base of IOQ)
      1006 |
           :
      1205 |
      1206 |  $101400 (SYSDB-relative pointer to base of Example Table)
      1207 |
           :
      5023 |
      5024 |  %123456 (20th word of IOQ)
      5025 |
           :
    102435 |
    102436 |  %654321 (30th word of Example Table)
    102437 |
           :
           :
end of SYSGLOB|____
```

Examples (a) and (b): Read DB+5 (=%1005) = %4000

Now, compute the effective address

E = %4000 + DB + X = %4000 + %1000 + %24 = %5024

Now, read the word at %5024 (%123456) and push it onto TOS.

(For case (b), substitute SYSDB for DB. SYSDB is always %1000.)

Example (c): Push 134 (=%206) onto TOS and execute an LST 0.

Read SYSDB + TOS (%1000 + %206) = %101400.

Now compute the effective address:

E = %101400 + SYSDB + X = %101400 + %1000 + %36 = %102436.

Now read (E) = %654321 and push it onto TOS.

Figure 2

Locating a Code Segment via the CST Entry

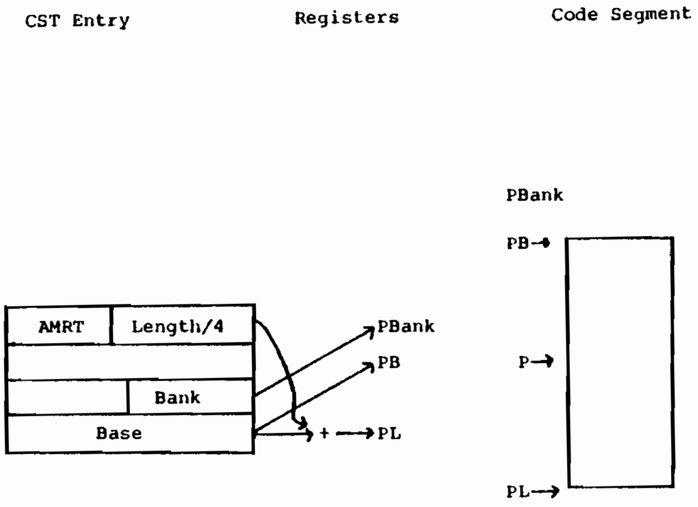
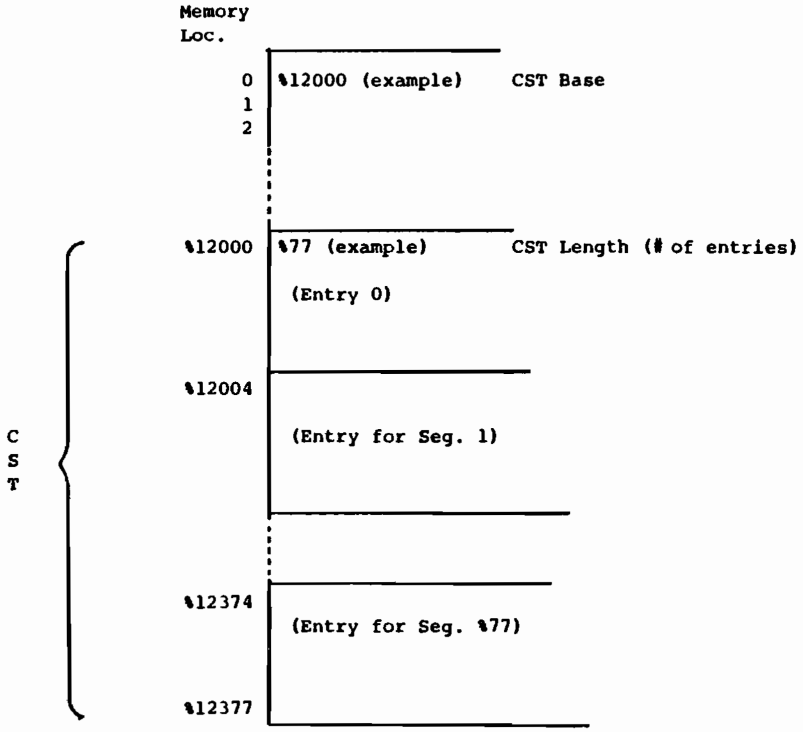

CST Entry
Registers
Code Segment
PBank
PB
P
PL
AMRT
Length/4
Bank
Base
PBank
PB
+
PL

Figure 3

Code Segment Table Structure

Series I and Previous Systems


Memory
Loc.
0
1
2
%12000 (example)
CST Base
%12000
%77 (example)
CST Length (# of entries)
(Entry 0)
%12004
(Entry for Seg. 1)
%12374
(Entry for Seg. %77)
%12377
C
S
T

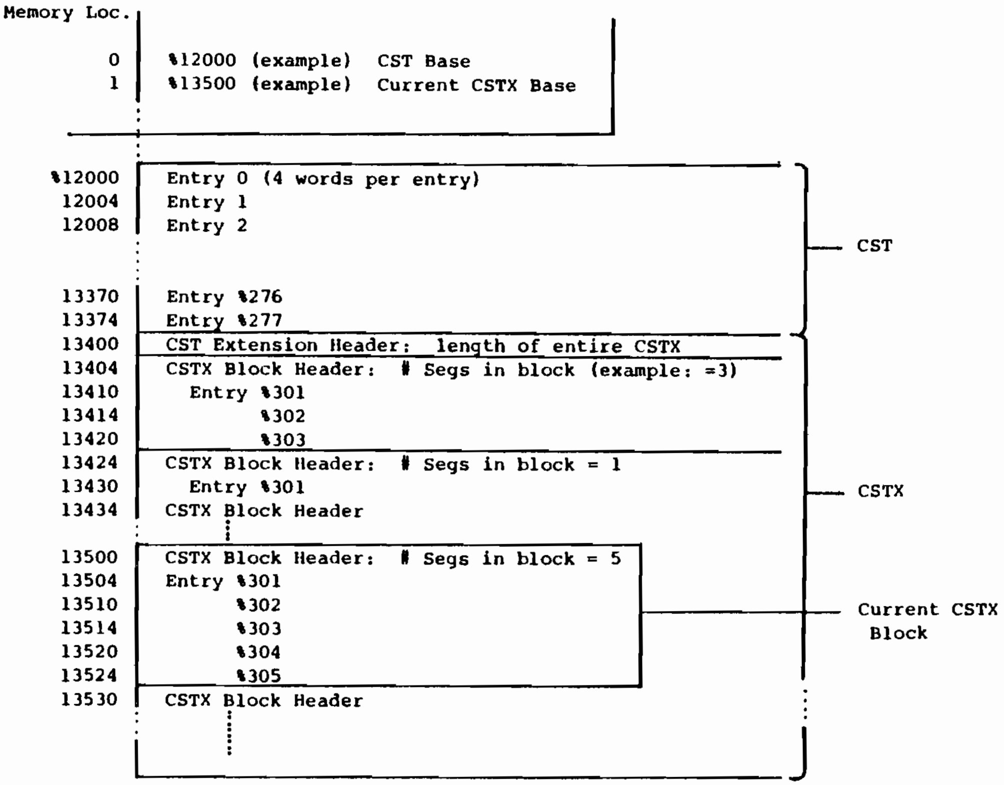
To find entry for Seg. 2, look at CST Base (%12000) + 2 * entry size

= %12000 + %10 = %12010.

Figure 4

Code Segment Table Structure

Series II through Series 64 (Pre-MPE V)



```
Memory Loc.

      0   %12000 (example)   CST Base
      1   %13500 (example)   Current CSTX Base

 %12000   Entry 0 (4 words per entry)                         ┐
  12004   Entry 1                                             │
  12008   Entry 2                                             ├── CST
    :                                                         │
  13370   Entry %276                                          │
  13374   Entry %277                                          ┘
  13400   CST Extension Header:  length of entire CSTX        ┐
  13404   CSTX Block Header:  # Segs in block (example: =3)   │
  13410     Entry %301                                        │
  13414           %302                                        │
  13420           %303                                        │
  13424   CSTX Block Header:  # Segs in block = 1             │
  13430     Entry %301                                        ├── CSTX
  13434   CSTX Block Header                                   │
                :                                             │
  13500   CSTX Block Header:  # Segs in block = 5             │
  13504   Entry %301                                          │
  13510         %302                        ──────────────────┼── Current CSTX
  13514         %303                                          │    Block
  13520         %304                                          │
  13524         %305                                          │
  13530   CSTX Block Header                                   │
                :                                             ┘
```

Figure 5

Logical Segment Transform Table(LSTT)

```
+-------------------------------+
| # of Logical Segments         |
+-------------------------------+
| Length of LSTT                |
+-------------------------------+  ---
| Physical Segment #            |
+-------------------------------+  logical segment 1
| Ptr to External Label List    |
+-------------------------------+  ---
| Physical Segment #            |
+-------------------------------+  logical segment 2
| Ptr to External Label List    |
+-------------------------------+  ---
|               .               |  .
|               .               |  .
|               .               |  .
|               .               |  .
+-------------------------------+  ---
| Physical Segment #            |
+-------------------------------+  logical segment n
| Ptr to External Label List    |    (max 255)
+-------------------------------+  ---
|M| STT #      | SEG #          |
+-------------------------------+  External Labels
|M| STT #      | SEG #          |  from Logical
+-------------------------------+  Segment 1
|               .               |  (if needed)
|               .               |
+-------------------------------+
|M| STT #      | SEG #          |
+-------------------------------+  ---
|               .               |  .
|               .               |  .
|               .               |  .
|               .               |  .
+-------------------------------+  ---
|M| STT #      | SEG #          |
+-------------------------------+  External Labels
|M| STT #      | SEG #          |  from Logical
+-------------------------------+  Segment n
|               .               |  (if needed)
|               .               |
+-------------------------------+
|M| STT #      | SEG #          |
+-------------------------------+  ---
```

Figure 6

Code Segment Table Structure

Series 42, 48, 68 (MPE V)

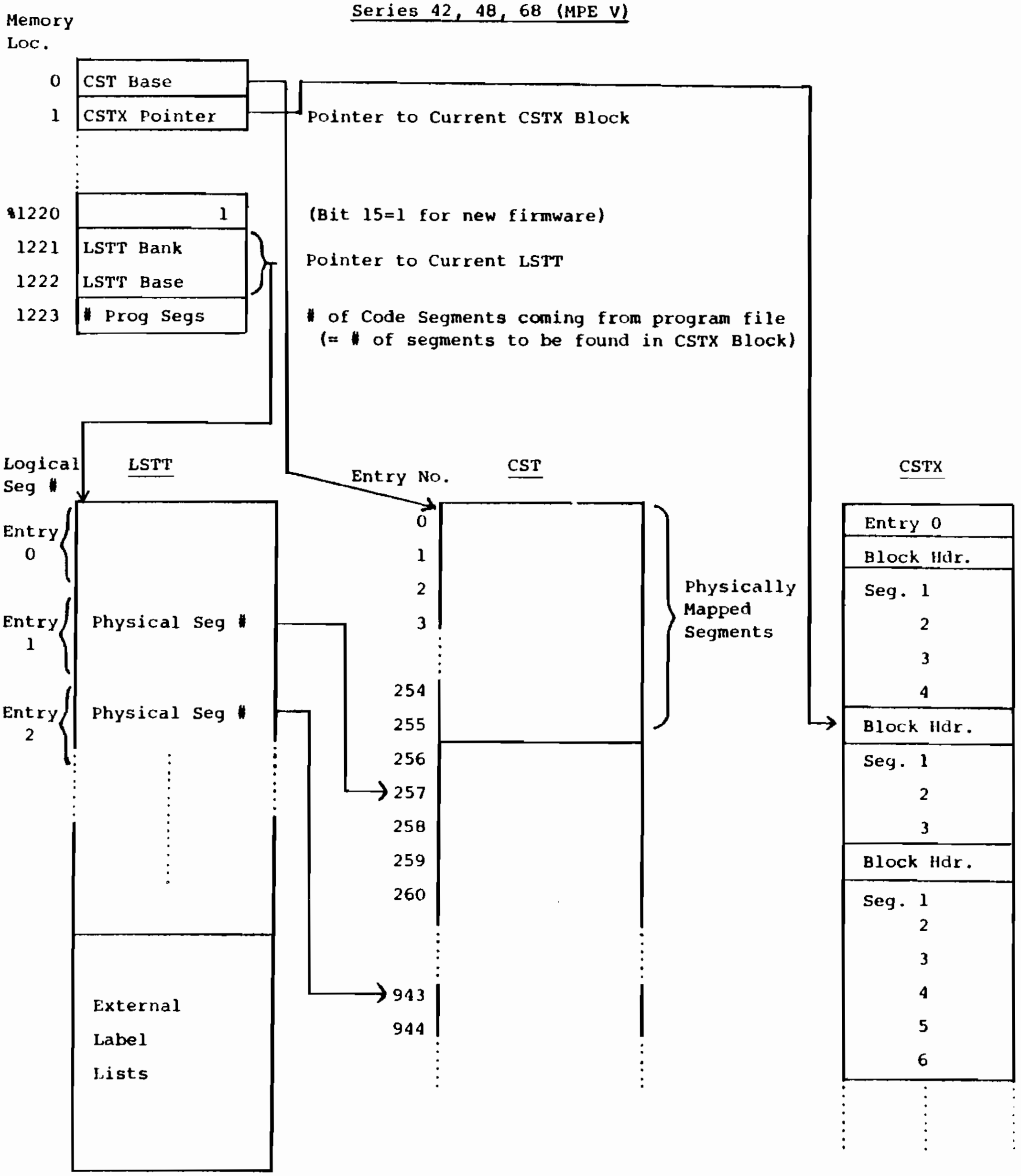

Memory
Loc.
0
CST Base
1
CSTX Pointer
Pointer to Current CSTX Block
%1220
1
(Bit 15=1 for new firmware)
1221
LSTT Bank
1222
LSTT Base
Pointer to Current LSTT
1223
# Prog Segs
# of Code Segments coming from program file
(= # of segments to be found in CSTX Block)
Logical
Seg #
LSTT
Entry No.
CST
CSTX
Entry
0
Entry
1
Physical Seg #
Entry
2
Physical Seg #
External
Label
Lists
0
1
2
3
254
255
256
257
258
259
260
943
944
Physically
Mapped
Segments
Entry 0
Block Hdr.
Seg. 1
2
3
4
Block Hdr.
Seg. 1
2
3
Block Hdr.
Seg. 1
2
3
4
5
6

Figure 7

Code Segment and Present STT Structure

```
     +---------------------------+
PB   |                           |
     |          Code             |
     |                           |
     |                           |
     +---------------------------+  --
     |                           |   |
     |                           |   |
     +---------------------------+   |  External
     |1| STT #       | SEG #     |   |  Labels
     +---------------------------+   |
     |1| STT #       | SEG #     |   |
     +---------------------------+  --
     |                           |   |
     |                           |   |
     +---------------------------+   |  Local
     |0|U|  Address              |   |  Labels
     +---------------------------+   |
     |0|U|  Address              |   |
     +---------------------------+  --
PL   |0|U|   0       |# of Labels |
     +---------------------------+
```

Bit 0 of the label designates whether the entry is an internal or external label. "U" designates whether the local label is callable or uncallable.

Figure 8

External PCAL Example

(Series II through 64)

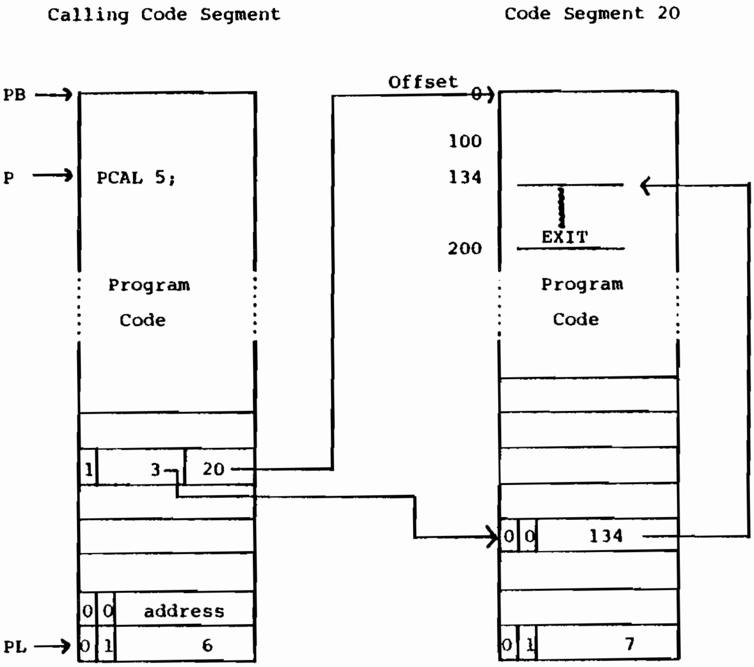

Calling Code Segment
Code Segment 20
PB
P
PCAL 5;
Program
Code
1
3
20
0
0
address
PL
0
1
6
Offset
0
100
134
200
EXIT
Program
Code
0
0
134
0
1
7

Figure 9

Code Segment and STT Structure for Logical Mapping

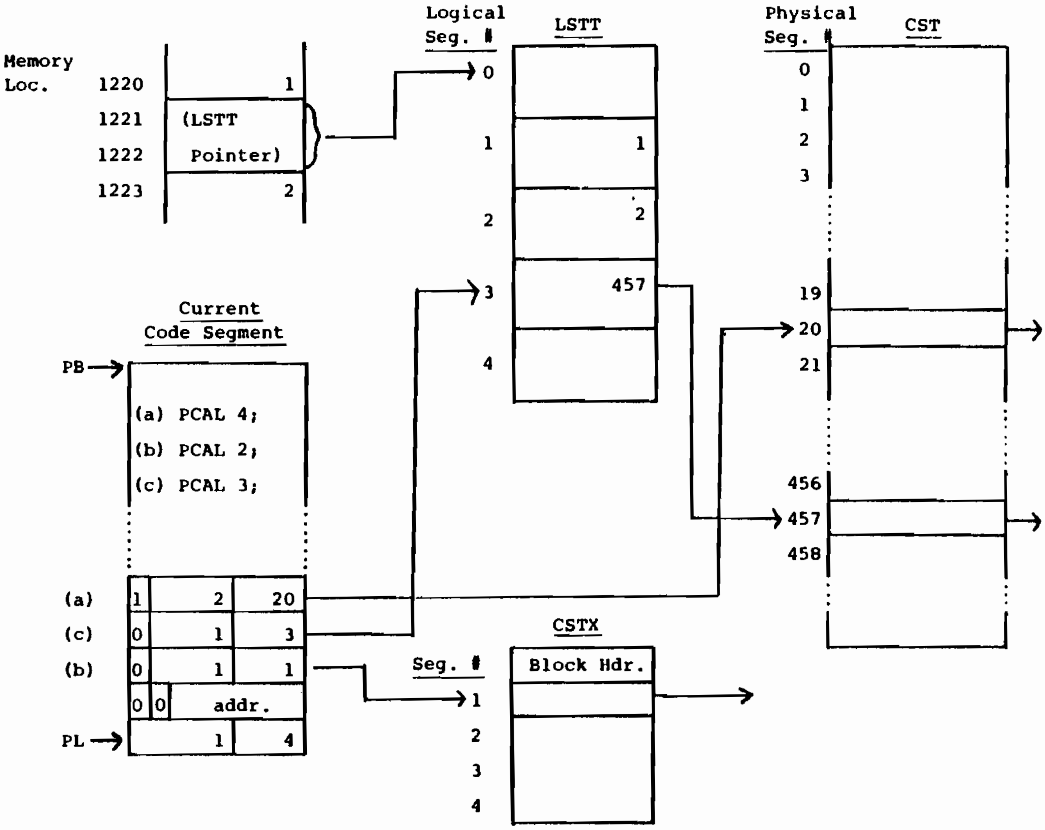
```
        +----------------------------+
PB      |                            |
        |          Code              |
        |                            |
        |                            |
        +----------------------------+  --
        |                            |  |
        |                            |  |
        +----------------------------+  |  External
        |M| STT #       | SEG #      |  |  Labels
        +----------------------------+  |
        |M| STT #       | SEG #      |  |
        +----------------------------+  --
        |                            |  |
        |                            |  |
        +----------------------------+  |  Local
        | |U|  Address               |  |  Labels
        +----------------------------+  |
        | |U|  Address               |  |
        +----------------------------+  --
PL      |#Local Labels|# of Labels   |
        +----------------------------+
```

Whether a label is local or external is determined by using the two counts at the head of the STT. "M" is used to designate whether the segment number in the label is a physical CST number or it is a logical CST number and must undergo a logical mapping through the LSTT to obtain the physical CST number (1=physical, 0=logical).

Figure 10

External PCAL Examples

(Series 42, 48, 68)


Memory Loc.
1220
1221
1222
1223
1
(LSTT Pointer)
2
Logical Seg. #
LSTT
0
1
2
3
4
1
2
457
Physical Seg. #
CST
0
1
2
3
19
20
21
456
457
458
Current Code Segment
PB
(a) PCAL 4;
(b) PCAL 2;
(c) PCAL 3;
(a)
(c)
(b)
1
2
20
0
1
3
0
1
1
0
0
addr.
PL
1
4
CSTX
Seg. #
Block Hdr.
1
2
3
4

(a) PCAL 4 transfers directly to STT #2 of Seg. 20, because this Plabel specifies physical mapping.

(b) This Plabel specifies logical mapping. Seg. # is 1, which is less than the number of program segments; so, this is a program segment, and we have to look at Entry 1 in the current CSTX Block.

(c) Since the Plabel specifies logical mapping, and 3 is greater than the number of program segments, we look at LSTT Entry 3 to find the physical code segment number; this is seen to be 457. We then look in the CST for Entry 457, which points to the code segment.

Figure 11

New Stack Marker

Q-3	X		
Q-2	T	M	ΔP
Q-1	MITROCEL	Seg #	
Q-0	ΔQ		

T = Trace or Control-Y interrupt pending

M = Physically mapped code segment

Figure 12

A Problem with Shareable Segments

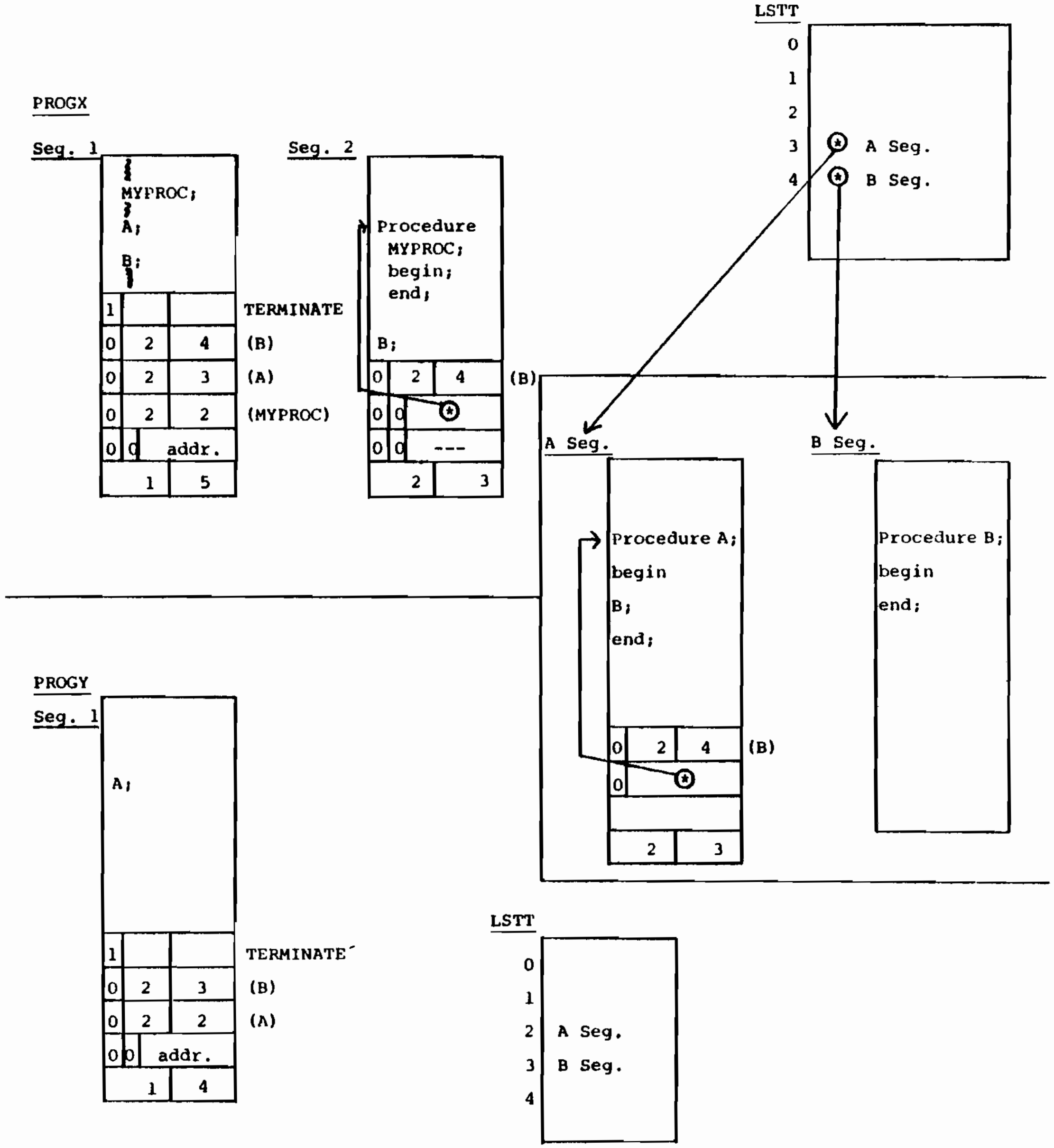
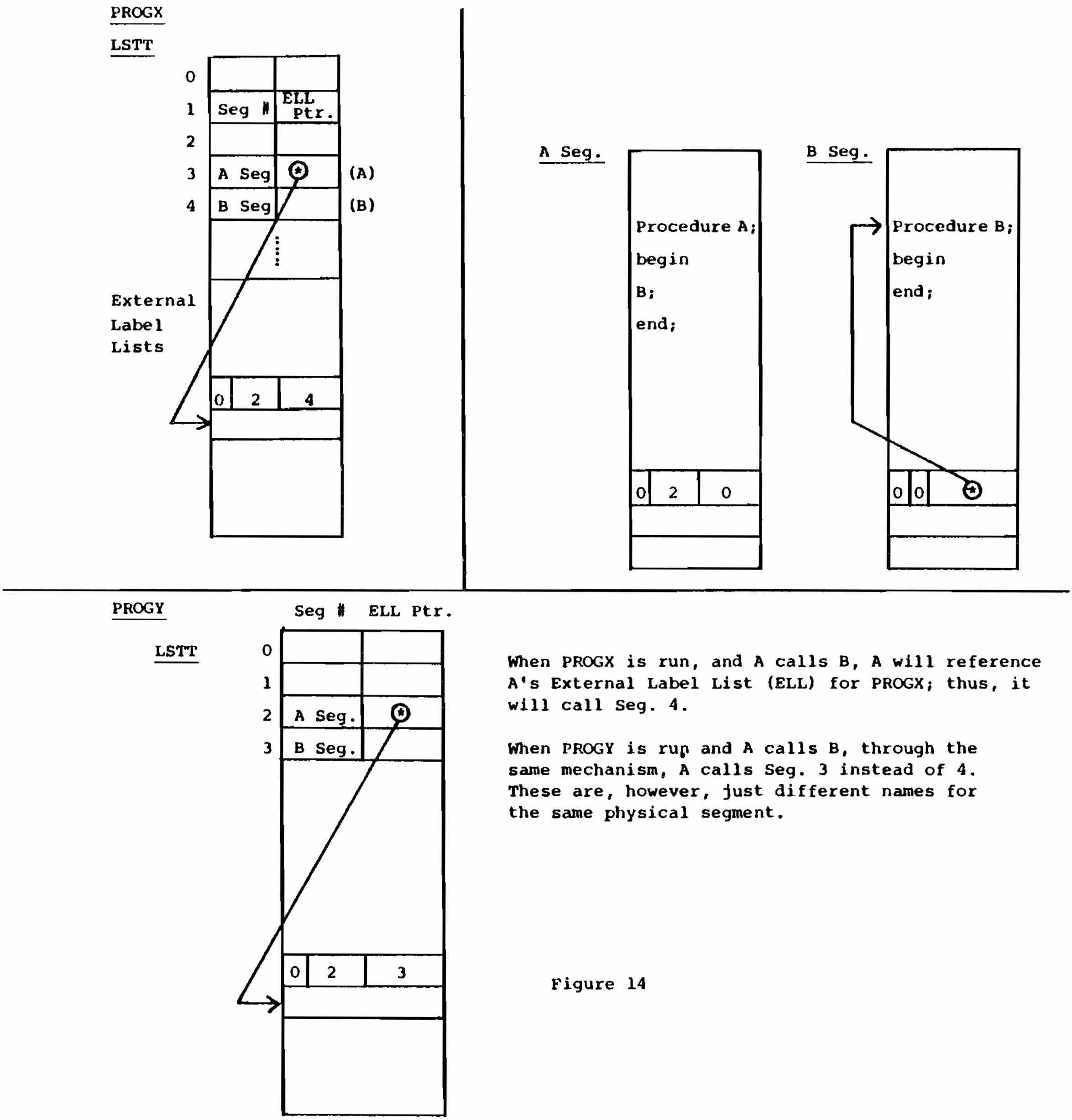

LSTT
0
1
2
3 A Seg.
4 B Seg.
PROGX
Seg. 1
MYPROC;
A;
B;
1
TERMINATE
0 2 4
(B)
0 2 3
(A)
0 2 2
(MYPROC)
0 0 addr.
1 5
Seg. 2
Procedure MYPROC; begin; end;
B;
0 2 4
(B)
0 0
0 0 ---
2 3
A Seg.
Procedure A;
begin
B;
end;
0 2 4
(B)
0
2 3
B Seg.
Procedure B;
begin
end;
PROGY
Seg. 1
A;
1
TERMINATE´
0 2 3
(B)
0 2 2
(A)
0 0 addr.
1 4
LSTT
0
1
2 A Seg.
3 B Seg.
4

Figure 13

Using the External Label List


PROGX
LSTT
0
1
Seg #
ELL Ptr.
2
3
A Seg
(A)
4
B Seg
(B)
External Label Lists
0
2
4
A Seg.
Procedure A;
begin
B;
end;
0
2
0
B Seg.
Procedure B;
begin
end;
0
0
PROGY
Seg #
ELL Ptr.
LSTT
0
1
2
A Seg.
3
B Seg.
0
2
3

When PROGX is run, and A calls B, A will reference A's External Label List (ELL) for PROGX; thus, it will call Seg. 4.

When PROGY is run and A calls B, through the same mechanism, A calls Seg. 3 instead of 4. These are, however, just different names for the same physical segment.

Figure 14

REMOTE SPOOLED PRINTERS IN AN OPERATORLESS ENVIRONMENT

by Tor Kristian Hånde
M. Sc., Eng. Ph.
Fjerndata
Norway

Abstract

This paper will present an overview of some of the problems often encountered at HP3000 installations that have a number of remote spooled printers.

We will look at the tasks that the remote operator is supposed to perform, and how the MPE spool-system helps or hinders him in carrying out his tasks.

One possible solution to many of the problems presented is that of automating some of the operator's tasks. We will present an implementation of such a "computerized operation" in this paper.

We will finally give some ideas on what we would like to see incorporated in such a "computerized operation".

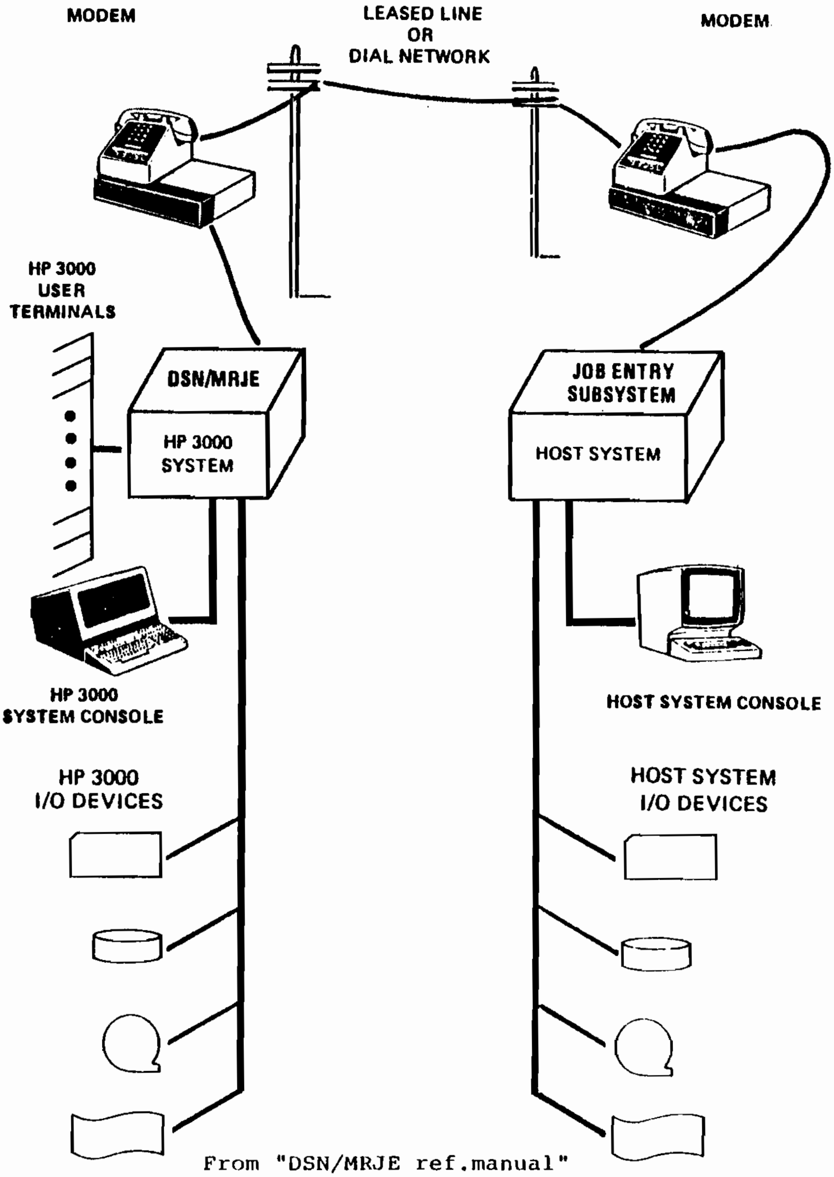
Introduction

When using DSN/MRJE or DSN/RJE a problem arises: The spoolfiles do not get printed where the user wants them, instead they all queue up for the same device.

The situation

One of our customers has a number of remote spooled printers connected to their HP3000. They also have access to our IBM 3081K through both DSN/IMF and DSN/MRJE.

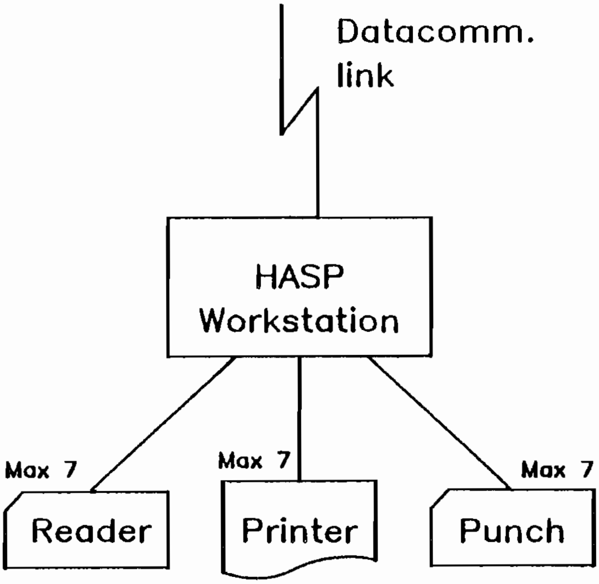
I assume that DSN/MRJE is the most used way of establishing batch access to an IBM mainframe. DSN/MRJE is HP's HASP Work Station Emulator. It is more sophisticated than DSN/RJE which emulates IBM 2780/3780. For this particular situation it does not matter which is beeing used. In fact, even if this problem arose using DSN/MRJE, it might as well have arisen from any other similar system. We will stick to DSN/MRJE here today. DSN/MRJE assigns a file-name to each spoolfile: The job-name from the mainframe. Furthermore DSN/MRJE puts the FORMS-code on the first line of the file, line # 0.


MODEM
LEASED LINE OR DIAL NETWORK
MODEM
HP 3000 USER TERMINALS
DSN/MRJE
HP 3000 SYSTEM
JOB ENTRY SUBSYSTEM
HOST SYSTEM
HP 3000 SYSTEM CONSOLE
HOST SYSTEM CONSOLE
HP 3000 I/O DEVICES
HOST SYSTEM I/O DEVICES

From "DSN/MRJE ref.manual"

Elements of a data communication system.

All remote spooled printers at this site have a unique device class name, generally with the form LPxx where xx is its logical device number. The users were requested to use this as



HASP Work Station

the IBM FORMS code for their listings in order to get them printed at the correct printer. MRJE wants to print all spoolfiles on the same printer.

Every time a forms request from DSN/MRJE with FORMS-code LPxx appeared on the screen, the operator refused to print the spoolfile and redirected it (ALTSPOOLFILE #Onnn,Dev=LPxx) to the correct remote spooled printer, namely the one with device class name LPxx. This also requires that the operator is present at the console all the time someone running DSN/MRJE is working. If I know DP personell right they work any time of the day and often far into the night (this was written around midnight). They also want their listing immediately.

As soon as this possibility came up, most of the users liked the idea of having their spoolfiles printed at the printer that was most convenient to them. This created a rush of spoolfiles to be redirected and the operator soon had his time more than fully occupied redirecting spoolfiles.

There is another aspect, the reliability was far from 100%. To quote Tom Gilb: "Any system which reliability is based on human reliability is unreliable." An operator needs breaks, takes lunch, mistypes etc. just as any other human beeing. This solution was only acceptable in an environment that received a small amount of spoolfiles only, where no one is in a hurry getting his listing (I have yet to see a such a person) and where no "touchy" data are transferred. At our customer's site it did not work as smooth as they wanted it to be.

The solution

At this instant we were asked to take a look at a way to automate this. The solution to the problem appeared to be rather simple, although not quite as simple as stealing candy from a baby. There is an excellent utility programme, SPOOK, that is far more powerful than the MPE-command ALTSPOOLFILE. It was tempting to try to use it.

For those of you that are not familiar with SPOOK, it is a utility programme that allows you to interrogate and manipulate spooled devicefiles. In brief you may alter a spoolfile's output device, its' priority, number of copies etc. You may also convert a spoolfile to an ordinary MPE-file. SPOOK lets you read files, purge files, merge files and some more manipulation. I'll advise you to take a look at it, you will find it in the Utility Manual.

Just letting the operator use SPOOK instead of ALTSPOOLFILE does not make things move the way we want them to. There is however a more elegant solution: let a program simulate an operator. SPOOK is executed as the calling programmes' son process and SPOOKs' standard input and output files are redirected to the father programme. This father programme, hereafter called SDS now sees SPOOK as two rather ordinary looking message files, one read-only and one write-only. If you then establish a table of forms-codes and file-mes and their associated outputdevices, you have all you need for automation of the work.

To make this work you have to configure a spooled device on your system that is not currently spooled (STOPSPOOL # n). This device will be the outputdevice for all spoolfiles from DSN/MRJE. This device need not exist, it is just a port that is not in use.

As the device is spooled and stopped, all spoolfiles will queue up very nicely and may be picked at choice, they're just waiting to be manipulated with.

After SPOOK is initiated we just simulate what an operator would do "by hand". At first we ask SPOOK to show us the spoolfilequeue by using the SHOW-command. SDS sees this as an ordinary FWRITE and SPOOK sees SDS as an operator and answers. We now read SPOOK's answer by an ordinary FREAD and look for the spooled outputdevice. As soon as we have found one we put the spoolfilenumber and filenumber in a small table for future use. When we found all the spoolfiles we are looking for, we text in each of them by using the TEXT- command and read the first line by using the LIST 0 command. We now have what we need for a comparision with our table of filenames,forms-codes and outputdevice. Now we're getting down to business: By using the ALTER-command with the outputdevice from our table as a parameter, we have now done what we wanted to do.

This table of matching file-names, forms-codes and outputdevices needs not be static. At our customers site this table is read once every five minutes and you change it simply by using the editor and write it back again. This reading of the file takes roughly half a second.

Controlling the SDS can be done at various levels of sophistication. One solution is no control at all, just let it run and terminate using brute force (that is :ABORTJOB nnn). That may work fine at an installation where it just runs forever. At a slightly higher level you may communicate using a message file opened in "nowaitio" mode and write your commands directly to it. The most sophisticated way of doing controlling it is by a programme that has a two-way communication through message files. This requires a command-parser (MPE intrinsic MYCOMMAND is well suited) and some more overhead. This is strictly not important and as it has little to to with SDS' operation, I will not go deeper into it.

Enhancements

A such a system opens the possibility for future enhancements. On top of our priority list is a way to run Remote Job Entry backwards. Instead of submitting a job from a terminal connected to a mini to be run on a mainframe, you may do it the other way around and have your listing routed back to the mainframe for further processing or whatever you like.

To be able to process a spoolfile as a jobstream it must be converted to an ordinary MPE-file. This to remove forms-codes and other info that DSN/MRJE generates. The same must be done if data is transferred the same way. When returning the result it must be reformatted. DSN/MRJE will only accept 80 coloumn input (guess where that comes from) and the listing is usually 132 characters wide plus an unprintable carriage control code CCTL that may be harmful and therefore must be coded in some way. This quite another story and we will not go deeper into it now. If anyone wants to hear more about our product that does that, we can arrange that.

Conclusion

The system as described here is already installed by at our customers site and he is well satisfied with it. In addition we use it ourselves in our laboratory.

I hope I have given you some ideas on how to improve automate the boring job of redirecting spoolfiles. As you will understand the principles outlined in this paper is not restricted to redirecting spoolfiles, but may be used in most similar situations.

Tor Kristian Hånde

MPE V: Product Overview, Project Development Strategy, and Implementation Methodology

Robert L. Mead
Project Manager

Hewlett-Packard Company
Computer Systems Division

Introduction

In addition to the external changes and new capabilities provided by MPE V, a number of substantial changes were also made to the internals of MPE. Three components of the MPE V release required significant development efforts as a direct result of the numerous internal changes necessary: support for larger configurations, expansion of the Code Segment Table (CST), and disc caching. The details of disc caching are described in another paper [1], and hence will not be discussed here. In this paper, we will concentrate on the changes in MPE V required for the support of larger configurations and the CST expansion. After a discussion of the changes to MPE internals, we will address the development methodology used for the project. This methodology differed from the traditional methods used within the lab, in an effort to shorten the development and testing cycle, increase productivity, and improve softare quality. Finally, we will discuss the strategy used during the implementation phase of the project.

Internal Changes to Support Larger Configurations

The vast majority of the development work for the release of MPE V will be unnoticed by most users of the HP 3000. The largest effort was for the support of larger configurations, which did not affect the externals of MPE to any substantial degree. This effort did, however, require significant changes to the internal structure of MPE. In particular, the format of many internal data structures required changes, and perhaps more importantly, the access methods to the data structures were altered.

The internal structure of MPE has remained essentially unchanged during the 10 plus years of its existence. Over the years, some areas have undergone changes as they were rewritten or extended, but there have been no major changes to a large number of data structures. Even with the release of MPE IV, a major development effort itself, the internal changes were primarily restricted to the memory management data structures. In order to expand the maximum configuration supportable under MPE, it became clear very early in the MPE V project that virtually every internal table or data structure within MPE would require some modifications.

The reason for these extensive changes dates back to the origin of MPE. At the time, the HP 3000's most precious resource was memory (the first system had a maximum memory size of 128KB). As a result, most data structures were designed to conserve space, necessitating the packing of one or more fields into a single word. This approach left as its legacy such MPE limitations as 256 processes (an 8-bit field so that two process IDs could be stored in one word) and 1024 DSTs (a 10-bit field). Over the years, as the processing power of the HP 3000 increased, it became possible (and desirable) to support more and more users on a single system. At the same time, the maximum memory size supported also increased dramatically. Finally, the limits imposed by the software data structures were being reached as the number of simultaneous users approached 100, with the PCB and DST limits most frequently encountered. In fact, in some cases where either extensive process handling was employed, or large number of files or data bases were in use simultaneously, these limits could be reached with as few as 50 or 60 users.

It is perhaps a tribute to the original designers of MPE that their creation survived as long as it did without a major redesign. In fact it is interesting to note that the interrelationships of the various tables and data structures were obviously well understood, since users seemed to be encountering the limits of many tables simultaneously. In any event, our primary objective with the tables expansion component of MPE V was to eliminate the software data structures as the limiting factor in the maximum number of supported users. As we investigated the number of table entries that would be necessary to support a large number of users (200 was our original target), it became apparent that just a redesign of the internal data structures would not be sufficient to increase the number of users supported. The real problem would be the size of Bank 0.

For those unfamilar with the concept of Bank 0, a simplified explanation follows: the original HP 3000 contained 128KB of memory, the maximum amount that can be addressed with a 16-bit word address. As it became necessary to increase the amount of memory supported, the expansion was accomplished by subdividing all of memory into "banks", each consisting of 128KB. This would allow the additional memory to be addressed simply by appending a bank number to the 16-bit byte address (the bank number portion of the address was maintained internally by MPE, and hence was invisible to user programs). There were, however, several privileged mode instructions that could not be easily modified to accomodate a bank number. These instructions were used primarily to provide efficient access to MPE internal data structures that were memory resident. To retain this efficiency, the first 128KB of memory (referred to as Bank 0) was ordained with a special status, in the sense that only data structures in Bank 0 could be accessed via these privileged instructions.

The data structures required to be in Bank 0 were those that were also required to be memory resident. These techniques obviously limit the cumulative size of the data structures, since the size of Bank 0 is limited to 128KB. It was this limitation that became the real challenge to supporting larger configurations under MPE V. A number of alternatives to overcome this problem were investigated, ranging from firmware and compiler modifications to a brute force modification of MPE. In the end, it was decided a firmware modification to two instructions was the best solution. The instructions involved were the Load System Table (LST) and Store System Table (SST), two instructions that allow quick access to system tables via a system level pointer. The firmware modification consisted of changing the pointer format from a 16-bit address in Bank 0 to a 5-bit bank number and 11-bit address, with the rightmost 5 bits of the address always assumed to be zero. One significant advantage to this approach was that it would be possible to execute the modified version of MPE on both the modified and unmodified firmware, as long as each table accessed via the LST/SST instruction is built on a 32-word boundary (since an embedded bank number of zero could be used with the unmodified firmware, and would be identical to the original pointer format). Once this approach was adopted, there were now 32 banks (or 4MB) of memory in which to place memory resident data structures.

It is important to note however, that even with the above firmware modifications, not all resident MPE data structures can be moved outside of Bank 0. Some tables, in particular the DST, CST, and several I/O related tables, are examined and or modified by instructions other than LST and SST. To minimize the impact on firmware, it was decided that these tables would remain in Bank 0.

The development effort required to support larger configurations thus consisted of two major components: (1) modification of data structures to accommodate larger table indices (and associated changes to MPE code); and (2) changes to the access methods used for system tables to ensure that they could reside outside of Bank 0. This second component involved making the LST/SST instructions the only access method for resident MPE data structures, and was necessary since in many places within MPE other techniques were used to access the resident data structures; techniques that would not work correctly once a data structure was moved outside of bank 0.

Internal Changes for Expansion of CST

The existing limitation on the size of the Code Segment Table can also be traced to the origins of the HP 3000 and MPE. However, in this case, the limits are more closely associated with data structures utilized by the firmware. The limit of 256 code segments accessable by a process is due to the availability of only 8 bits for the segment number in the status register. This 8-bit field is also used in the Segment Transform Table (STT) and in P-labels, packed with a 7-bit STT offset and a single bit flag. Without extensive revisions to the firmware and MPE, it was virtually impossible to extend this limitation.

The limit of 256 segments is further subdivided into two categories of segments: segments 0-191 (%0-%277) are used for segments from segmented libraries (including MPE), and segments 192-255 (%300-%377) are used for program file segments. Since entries 0 and 192 (%300) are both used to contain header information in the associated data structures, we are left with the existing limits of 191 sharable SL segments and 63 program file segments. The 191 limit refers to the combined total of SL segments referenced by all active processes.

The approach adopted to increase the number of available code segment was to slightly alter the concept of the two categories of code segments, but with one important difference: both categories would consist of 256 segments. The trick of course was to devise a method to distinguish between the two categories, denoted as physically mapped segments and logically mapped segments. The obvious choice was to use a single bit flag. The difficultly was in finding a "spare" bit in all of the data structures that dealt with code segment numbers. The status register was handled by using a bit in a different hardware register to indicate logical/physical mapping, and the bit was stored in bit 1 of the delta-P word of the stack marker (since code segments are limited to 16KW, at most 14 bits are needed for delta-P anyway). The P-label format and STT proved to be a bit more challenging, and eventually required a redefinition of a bit in the existing data structures. The change to the STT involved redefining the internal/external bit to indicate logical or physical mapping. The internal/external resolution was handled by inserting the number of local (internal) entries in the header word of the STT, and then always allocating the internal entries first (as it turned out, this was already the case). Finally, the P-label format was also modified to use the internal/external bit for the logical/physical distinction. Unfortunately, this implies that it is no longer possible to execute a PCAL 0 instruction with a local (internal) label on top-of-stack.

Now that we have discussed how the logical/physical distinction is made, it is appropriate to define what it means for a segment to be logically or physically mapped. Physically mapped segments are always allocated from the beginning of the CST table, and are reserved for system SL segments (i.e. that is primarily MPE). If a segment is physically mapped, the segment number is directly mapped into the CST table. Logically mapped segments can be either program file segments (residing in the CSTX) or user SL segments (residing in the CST). In order to determine where a logically mapped segment resides, a new data structure is now associated with each process, the Logical Segment Transform Table (LSTT). The LSTT contains information about each logically mapped segment associated with a process, and each LSTT consists of one data segment. It should be noted that the code sharing functionality of MPE and the HP 3000 has been retained despite these changes, since processes executing the same program file will share LSTT segments, and the same CST entry for a user SL segment can be pointed to by multiple LSTTs. The LSTT segment associated with a process must be in memory whenever the process is executing however, since some instructions must be able to access information in the LSTT. In addition, the total number of CST entries for the entire system has been increased to 2048 (from 192), although any individual process can only access 255 of the CST entries.

It should be obvious by now that the changes described above involve firmware changes, as well as changes to MPE. Modifications were required in a total of eight instructions: five in the standard instruction set (PCAL, SCAL, LLBL, EXIT, IXIT) and three in the extended instruction set for support of COBOL II (XBR, PARC, ENDP). The functionality of these instructions was not changed, only the internal operation. On the software side, a number of changes were required. The MPE loader was essentially rewritten to accommodate the concept of logically mapped segments and to create the appropriate mapping segments as programs were loaded, changes were made to some memory management procedures to ensure that the mapping segment is always available when a process is executing, and several of the privileged debug commands were enhanced to allow specification of both logically and physically mapped segments. It is important to note however, that the modified software can be used on a system with or without the modified firmware. The inverse is not the case however.

This discussion has provided a brief overview of the internal changes to MPE that were necessary to provide support for expansion of the CST table and for support of larger configurations. Additional details concerning the architectural changes are provided in a paper discussing the firmware changes for MPE V [2]. We will now turn our attention to the development methodology employed for the MPE V project.

Development Methodology

As hopefully is evident from the above discussion, a substantial effort was required to produce MPE V. The development phase for the large configuration support alone involved in excess of twenty engineers over a period of approximately 7 months, although not all engineers were involved for the entire period. The coding changes required either modification or examination of a significant percentage of the approximately half million lines of code in MPE. Similar efforts in the past have required as long as two years to develop and release. In the case of MPE V, our objective was to complete the project and release the software in nine months.

The development effort for MPE V involved numerous changes to the internal data structures used by MPE. As it turns out, MPE was not the only software on the HP 3000 that directly accessed MPE internal tables, or that had packed several fields into a 16-bit word. IMAGE/3000 for example, internally used a 10-bit DST index in conjunction with a 6-bit file open count in a single word. The expansion of the DST under MPE V thus necessitated a change to an internal IMAGE data structure, as well as the corresponding coding changes. IMAGE also directly calls internal privileged procedures within MPE. The parameters to some of these procedures, as well as the calling sequence, were changed with MPE V, also requiring changes to IMAGE. Almost all data communication products were also affected, in most cases to an even greater degree than IMAGE/3000. In particular, changes to the structure of the I/O related tables required coding changes to data communication drivers and low level procedures.

In order to mimimize the elapsed time to complete development of MPE V and the associated subsystems, a different approach was taken during the development phase of the project. In the past, the project groups responsible for MPE, data communication products, and application subsystems worked more or less independent of each other. In general, each team would do their development using the previous version of the other products, and then any conflicts introduced by changes and enhancements to one of the products would be resolved during a final integration and testing phase. The integration would begin after a coding freeze for all products involved in a given release. Unfortunately, when the changes to one of these products were extensive, this integration phase could be quite lengthy. This was especially true if MPE had undergone any major internal changes.

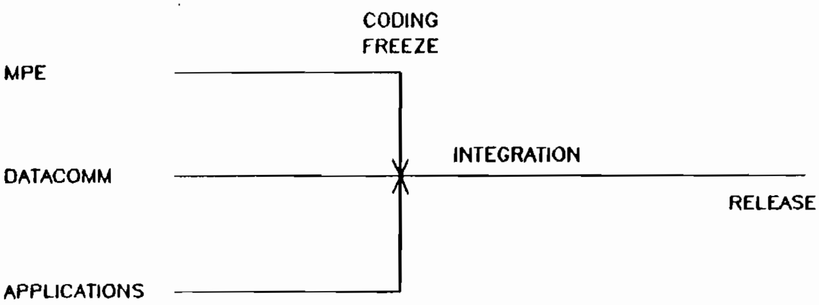


FIGURE 1 - TRADITIONAL DEVELOPMENT CYCLE

This traditional approach to development is depicted in Figure 1. The primary advantage to the traditional approach is that it minimizes the development resources for each team, since they are always using a known good version of the other products. It does however increase the liklihood of a long integration cycle, particularly when one or more products changes substantially.

Since it was obvious from the beginning that the changes to MPE for MPE V would have a major impact on other software products, a new approach to the development phase was adopted. This approach, depicted in Figure 2, allowed the data communication and application teams to be exposed to the changes in MPE early in the project, instead of a waiting until the coding freeze that typically occurred just prior to the marriage of the various software components. This phased development approach was expected to be more successful than the traditional approach. At each phase, a working (although not fully debugged) version of MPE V would be provided to the data communications team, who could then begin to test the modifications to their software against the modified version of MPE.

To further reduce the development time, the data communications team produced the necessary coding changes for their products while the MPE team was testing their changes. This was not without some disadvantages however, since the data communications team would be working with a version of MPE that was likely to contain problems, and hence would slow their efforts. This approach required close cooperation between the two teams to ensure that any problems encountered would be quickly resolved. Once the data communications team had successfully integrated and tested their changes to accommodate the internal data structure changes, the combined MPE and data communications products were passed on to the applications team for their testing. In this way, there would be no surprises when all products were officially integrated prior to the release of the software.

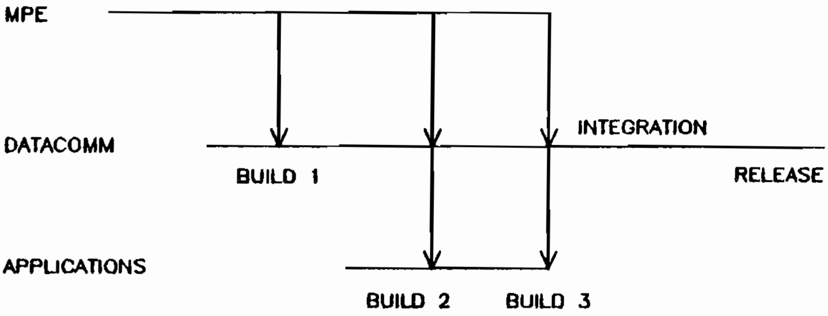


FIGURE 2 - MPE V DEVELOPMENT PLAN

The types of changes required to produce MPE V turned out to be almost ideally suited for this type of approach. If done properly, it was feasible to make the modification for a small subset of tables, and test those changes, prior to making the changes for other tables. The only requirement was that once a table was converted to its MPE V format, all code that dealt with that table must be using the new format. It was not necessary to convert all tables to their MPE V formats simultaneously.

With this goal in mind, the set of tables and data structures to be modified were subdivided into 3 builds (or phases). The first build was primarily oriented around the changes required to the I/O system tables. This set of tables was selected to be converted first since they had a major impact on the low level data communication code, and thus completing them first would allow for maximum exposure and testing. Build 2 was intended primarily to include the data structures involved with the file system and process handling. These changes had a large impact on the higher level data communication software, and once again the goal was to maximize testing exposure. The third and final build included the job/session related data structures, as well as any other items not included in builds 1 and 2. It was not until build 3 was complete that it would be possible to actually expand any of the tables beyond their MPE IV limits.

Implementation Strategy for Changes to MPE

In addition to structuring the development phase to allow for early integration of data communications and applications software, the changes to MPE itself were also planned to maximize the early testing of the software. It would have been easy to simply make all of the required changes for one or more MPE data structures first, and then spend a long time debugging this rather massive set of changes. Instead a more organized (and hopefully more productive) approach was adopted.

At the source level, MPE is comprised of approximately 100 modules, each resulting in a system program file, or one or more SL segments. These modules are organized around the functional components of MPE, such as the file system or kernel. In the past, each module contained its own internal definitions of the data structures it accessed, even when the data structures themselves were accessed by multiple modules or functional components. In an effort to make MPE more maintainable, as well as to minimize the development time for MPE V, a significant effort was undertaken in the initial phase of the project to develop a set of common definitions for the various data structures; these definitions to be used by all modules accessing a particular data structure. This was accomplished by using the $INCLUDE file feature of the SPL compiler.

The initial step was to create a set of include files describing each data structure in its MPE IV format. Once these were complete, modification of the various modules could commence. Two changes were made at this first step: (1) the modules were changed to use the common include files; and (2) all changes required to address tables residing outside of bank 0 were made. This second type of change, to provide MPE V addressing of tables, was facilitated by the use of the common include files, but still comprised the most difficult step of the project. At this point all tables were still in their MPE IV formats however, which allowed for extensive module testing prior to integration of all modules. The system produced from this integration was dubbed MPE IV.5, since it contained a combination of modules using MPE IV table formats and MPE V addressing techniques.

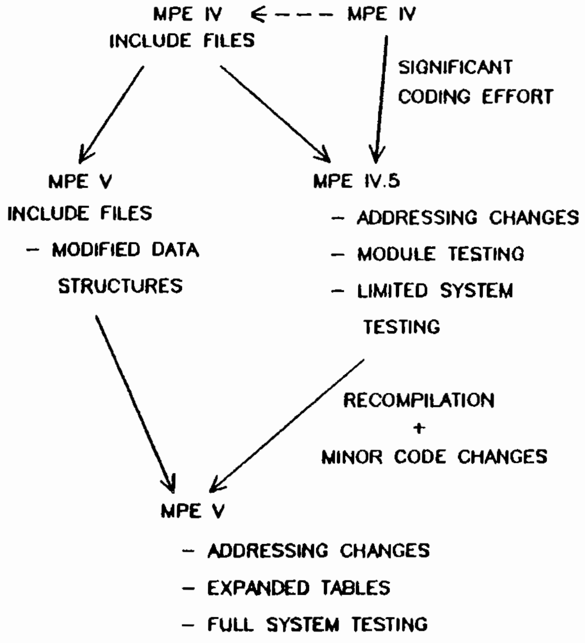

MPE IV INCLUDE FILES
MPE IV
SIGNIFICANT CODING EFFORT
MPE V INCLUDE FILES
- MODIFIED DATA STRUCTURES
MPE IV.5
- ADDRESSING CHANGES
- MODULE TESTING
- LIMITED SYSTEM TESTING
RECOMPILATION + MINOR CODE CHANGES
MPE V
- ADDRESSING CHANGES
- EXPANDED TABLES
- FULL SYSTEM TESTING

FIGURE 3 - MPE V IMPLEMENTATION STRATEGY

In parallel with the coding changes to produce MPE IV.5, a separate set of common include files were developed. These include files defined the MPE V data structure formats, and were derived from the MPE IV include files with a minimum of effort. Once the modified include files were available and the MPE IV.5 version of a module was functional, an MPE V version of the module could be created simply by recompiling the module with the MPE V include file. Minor coding changes were also required in some cases. The resulting modules when integrated together would produce an MPE V system. System level testing could then be conducted. While not all coding errors were detected at the module testing stage, the time required to integrate and successfully test the resulting system was significantly less than what had been the case on previous projects. This strategy is depicted in Figure 3, and was used for each of the three phases of the project.

Summary

MPE V represents a substantial development effort, due to a major redesign of many internal MPE data structures. This redesign was necessary to expand the capabilities of MPE and the HP 3000 with regard to the number of users that can be supported. Due to the nature of the project, and the need to make these expanded capabilities available to the customer base as quickly as possible, a new development methodology was employed. This methodology involved a phased implementation strategy that provided for earlier exposure of the modified version of MPE to other HP software products than has traditionally been the case. As a result, the development phase of the project was significantly shorter than previously had been the case for a project of this magnitude. At the time of this writing, the final integration and testing of MPE V was not yet complete, although indications are that this phase will also be much shorter than experience would predict.

[1] BUSCH, John R., and KONDOFF, Alan J. MPE Disc Cache: In Perspective. Proceedings of the HP 3000 International Users Group, Edinburgh, October 2-7, 1983.

[2] HOLINSTAT, David. Architectural Changes for MPE V. Proceedings of the HP 3000 International Users Group, Edinburgh, October 2-7, 1983.

EFFECTIVE SITE PLANNING STRATEGY

Lynn K. Darnton
EG&G Instruments Ltd.
Bracknell, U.K.

INTRODUCTION.

"Adequate and proper site preparation is vital to the extended reliability of your computer system" (Hewlett Packerd - the Site Planning and Preparation Guide, hereafter called 'the Guide').

The above statement is true, regardless of the size of the installation. The Guide is a very comprehensive document, containing all the technical specifications required for the design and construction of the perfect environment for your new HP3000 computer. In great detail it explains planning principles, site selection, raised floors, air conditioning, fire prevention and detection, electricity supplies, safety, delivery, data protection and a wide range of other subjects.

However, in the area of local building, fire and safety regulations, the Guide is necessarily vague, merely pointing to the fact that regulations exist and that they must be checked. It would be impossible for Hewlett Packard to include in any manual the totality of laws, rules and regulations. They vary considerably from place to place and are subject to change from time to time. The reader is warned that this whole area of law and local regulations can be complex and misleading.

For example, paragraph 2-16 of the Guide says ".....be sure to consult local building codes for fire prevention and protection.". In this case, in certain geographical regions, local building codes may demand that all walls, doors and partitions be "half hour fire rated" minimum; let us say the correct fire-rated doors, walls and partitions are erected and the system eventually goes live. During a routine Fire Department annual inspection, the Company's Fire Certificate is withdrawn because the computer room (classed as a 'fire generation room') should be separated from the adjacent fire hazard areas by minimum one-hour fire rated walls. The Fire Department will issue a Fire Certificate only when the offending walls have been replaced or reinforced ! Meanwhile the Company is not permitted to trade without a valid Fire Certificate !

Furthermore, compliance with the material requirements of some regulations may not be sufficient to satisfy some authorities; for example, a structural extension may have to be dismantled because, although it satisfies the requirements of all current building regulations, planning permission was not sought, but is required.

Having traced all applicable laws, rules & regulations, the non-lawyer DP person may be baffled by the complexity of legal jargon. So, it is very easy to say "read the Guide in conjunction with all regulations currently in force", but having read all the regulations, he or she may be non the wiser.

Where does the average DP person begin and how does one control a complex planning and construction project ?

In this paper I consider the problems of those who are engaged in the second (and most common) of two broad classes of planning project:

i. the construction of a brand new facility from the ground up, or

ii. the modification of an existing building or office.

I assume that all requirements for hardware, media, consumables, furniture, ancilliary equipment, telephones, staffing levels, etc., have already been defined and that our task is the design of the computer suite itself.

GETTING STARTED - THE SITE CO-ORDINATOR.

A Site Planning Co-ordinator should be appointed to take complete responsibility for the entire planning and construction project. This person may not be called "Site Planning Co-ordinator" by job title, but everyone on the project and all external persons, must recognise the co-ordinator's authority and role. No action should be considered legal unless authorised by the co-ordinator.

The co-ordinator will have a huge task involving many alien disciplines and it is unlikely that all will be understood sufficiently by any one person within the project timescales.

The co-ordinator will be a highly self-motivated person who can live under considerable stress, adapt to any required task quickly, communicate with all levels of staff from labourers to the M.D., possesses some tact & diplomacy. Ideally, (s)he will have an operations background and a wide knowledge of DP generally, gained through either a good acadamic course or long and varied experience. The installation's intended HP3000 System Manager is a good choice.

THE SITE PLANNING TEAM.

A Site Planning Team should be formed, consisting of the DP or Management Services Manager and Co-ordinator as the minimum nucleus. The team may be smaller or larger depending upon the organisation size. Different persons will be drafted onto the committee for specific reasons as and when the need arises.

Transient members may include user department Managers, suppliers, consultants, technical specialists, union officials, etc. Hewlett Packard account engineers should hold permanent committee positions, as they should always be aware of the current project status and can be a source of expert consultation.

The committee, or team, should be an 'organic' entity, changing its membership as the project progresses and should not be seen to hold any authority in itself - the committe is a consultative tool and a communication medium.

Before embarking on the actual site planning sequence, I would like to discuss some fundamental behavioual rules and project management tools which you will find useful:

THE HUMAN FACTORS:

Never forget that one of the most important components of the new computer system, yet one of the most sensitive, is the human being.

Many computer projects will displace or relocate staff, especially where the Company has never had a computer before. Like cats, people often hate and resist sudden change. Most humans suffer a deep rooted fear of being replaced or controlled by machines, robots, etc. Where a new DP Department moves into more than one room, two or more previously separate departments will find themselves sharing one office; departmentel social structures and heirarchies will be destroyed, causing some degree of disorientation.

If several departments must be joined under a common roof, try to select job-related people, for instance invoicing & credit control or purchasing and cost.

When selecting which room(s) to convert to the new computer suite, try to select those presently occupied by future users and preferably users who will benefit most from the new system. In this way, their disturbance does at least have some identifiable, quantifiable pay-off.

During the feasibility survey and design phases of the computer project, staff should have been motivated to support the new system. This effort must be maintained throughout the site planning & construction phases. Support for the Company objectives must be reinforced. Users and displaced staff as well as tradesmen and construction workers can be asked to contribute their ideas; good ideas and information can be seen to be used, so that the construction work is seen to be a group user effort, not exclusively the effort of the new DP Department.

Resistance to change will decline as users identify the construction project as their own; during the completion of various sub-tasks, individual users or departments should be thanked for any contribution made, however trivial. People who resisted the new system initially, will eventually become the proud creators of their Company's new computer system. When they eventually use the system, they will be using something which is partly their own creation.

DOCUMENTATION:

Throughout the planning and construction project, several tasks will run parallel; each will require different resources & lead times and will be subject to varying degrees of Murphy's Law (ie. delay !). All this activity and unforeseen change must be monitored and controlled in some way.

Even when the project is being conducted entirely by one person, several tasks will be at various stages of completion or lead time at any moment in time.

The site planning and construction project is controlled as a computer design project in its own right. Each phase from initial surveys through to installation is recorded in detail.

At completion of the project the Co-ordinator should have produced, as a by-product of the project, a Company Installation Manual. This manual will contain ground plans for the Company premises, computer suite, structural details of floors, walls, ceilings, electricity supplies, alarm systems, plumbing, air conditioning, fire detection and fire fighting systems, telephone and data communications circuits, computer system cabling, local terminal network cabling, heat load maps, floor weight distribution, etc. In other words, a complete map of every part of the computer suite. The computer environment is seen as a complex system in its own right, and must be serviceable by your successors.

Parts of the Installation manual can be copied and distributed to an in house maintenance department , Hewlett Packard Customer Engineer, Fire Department, British Telecom, Air Conditioning Vendor, Computer Operator, etc. Each person receives only that part of the manual which pertains directly to his or her functional relationship with the DP installation.

When future changes are made to the hardware configuration or to the structural or environmental specifications or layout, the Installation Manual should be updated in just the same way as you expect software suppliers to update their manuals.

If some plumber cannot find a pipe, or an electrician can not find the air conditioning condenser manual circuit breaker, when either of these items require servicing twelve months hence, either the Co-ordinator did not document them, or someone did not communicate to the Co-ordinator.

During the project discussions will occur and agreements will be made. The latter will often be verbal, said quickly, under stress. Information will be passed down chains of people and through the Co-ordinator from source to destination. People will also disagree about what was said. Discussions should be recorded and filed. There should be little distinction between letters and the spoken word, especialy in matters which could lead to a major purchase or some other form of contract. The award of damages or the replacement of defective or unsuitable equipment could hang on what was said between two people on the phone months previously.

Engineers and consultants from several specialist and expert companies will each supply conflicting information about major design points. The fact is that many experts, like yourself, have picked up much of their expertise by working at a job under supervision - an apprenticeship type of situation. Defective knowledge will be accepted and used, because no one proved it to be incorrect. Techniques learned many years ago on one project may be assumed to be the global rule or method; these same techniques may then be applied to other projects where conditions actually require a completely different method. Because practical (but very narrow, specialised) experience may not always be backed up by theoretical knowledge,an expert is sometimes revealed to be an expert in very few circumstances. In other words, let each person do his or her work to the best of their ability, but you ensure that your objectives, discussions, agreements, contracts, observations, etc, are all very well documented.

See Exhibits 2 through 5 - Discussion Records.

BAR CHARTS

Bar charts can be very useful for illustrating planned sequences of tasks which are not very complex. Planned and actual timescales can be shown.

In the figure below, shaded boxes represent planned timescales for each job, while the solid bars indicate the actual times taken.

Depending on the materials upon which the bar chart is constructed it is possible to incorporate a 'cursor', or movable vertical bar, which shows the state of current jobs.

Bar charts can be constructed with any number of rows to depict extremely complex sequences, but this form of chart does NOT show the RELATIONSHIPS between jobs.

In other words, it is not possible to evaluate, using a bar chart, which jobs interface with which other jobs, or to see which jobs must be completed as a pre-requisite for other jobs beginning.

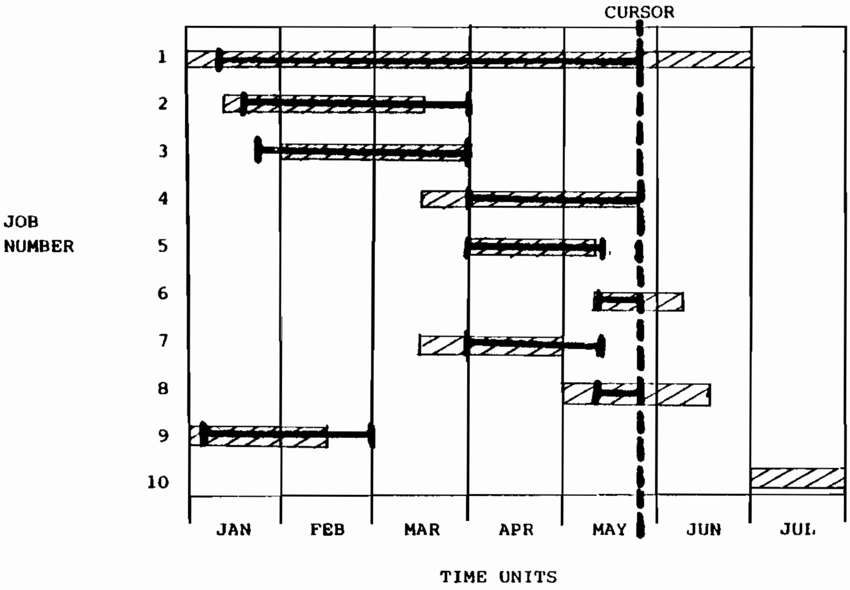


FIG. 1 - BAR CHART EXAMPLE

If it is necessary to illustrate the relationships between interfacing jobs, then the network diagram is more appropriate.

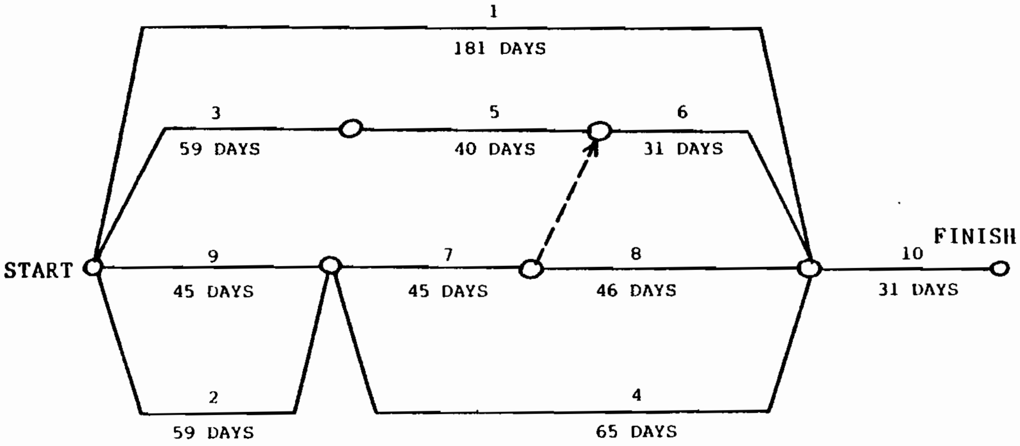
NETWORK ANALYSIS.

Network analysis is an extremely useful production planning and control tool. During any major project, a number of activities proceed separately, in parallel. One or more activities must be completed as a pre-requisite for one or more other activities to commence. This complicates resource scheduling and overall project planning, unlike a situation where each separate project task merely follows in line, sequentially. In the latter case, the overall project time would obviously be the sum of all times for all tasks.

If all tasks could start at the same time and run parallel, the overall project time would obviously be that of the longest task.

Where neither of these simple models apply, but there exist many more complex relationships, a network can be constructed. Below is a simple network depicting the inter-relationships between the various jobs previously illustrated in the bar chart (Fig. 1).

FIG 2 - NETWORK DIAGRAM EXAMPLE



Lines represent activities, while circles represent events, or the start or finish of one or more activities. The earliest & latest possible start and finish times can be written into each event.

It is obvious from this simple network, that the longest path is through tasks 1 & 10. Initially, this is the only path on which time savings can be made. This path controls the eventual finish date of the entire project; it is thus called the critical path. The critical path may well change as individual activities take longer or shorter than the scheduled timescales.

'Dummy' activity 7.1 shows that activity 6 cannot start until 5 & 7 have both finished, but that activity 8 is dependent only on 7 finishing.

PROGRESS CHARTS.

In addition to bar charts and network analysis, it is very useful to record events which occur during each activity.

Exhibit 1 illustrates a 'progress chart'. A sheet of paper is simply divided up into as many rows as there are things to monitor in this case telephone lines. As few as possible columns are drawn in to hold constant data. The remaining columns are assigned to dates, either periodical such as weekly, or ad hoc.

Table entries can be made in text form or as references to discussion records or both. The table shown permits the co-ordinator to survey quickly the situation on any telephone line or on all telephone lines at any date, or what was last said about a particular location.

Progress charts can be varied to record any large collection of related information. They could be used, for instance, to monitor costs. Planned items of purchase could be listed with budget costs. Each phase including purchase order, delivery, invoicing,payment and any exceptional conditions could be noted for each item. A variance column could provide up to date information on present cost status.

The progress chart can be used as a matrix to visually tie up a file of discussion records. In the case of a disagreement, it is possible to identify quickly in which sequence people were interviewed, then identify the discussion records pertaining to the subject of the disagreement.

Exhibits 2 through 5 show typical discussion records. They are almost direct copies of actual discussions. Names of companies and individuals have been altered by request.

EXHIBIT 1 - PROGRESS CHART

INSTALLATION PROGRESS - PRIVATE CIRCUITS & PSTN DIAL BACK-UP LINES

LOCATION	TYPE	JOB NUMBER	CIRCUIT NO.	SCHEDULED	SCHEDULED DATE / PROGRESS AS AT				
			TEL. NO.	DATE	20.10.81	27.11.81	30.11.81	1.12.81	2.12.81
A END	LEASED							INSTALLED	
	DIALUP	VJTNNN-NNN				INSTALLED			
LIVERPOOL	LEASED	9999/M/YY	Rnnnnnn		INSTALLED				
	DIALUP	AAA999/999	999 9999 999 9999					INSTALLED	
LONDON	LEASED	9999/M/YY	RLSW99999					OVERALL LINEUP 2.12.81	LINEUP FAILED - SEVERAL WEEKS DELAY
	DIALUP	AAA999/999	999 9999 999 9999				1.12.81	TODAY	OFFICE NEEDS REWIRE NO DATE
EDINBURGH	LEASED	9999/M/YY	Rnnnnnn					22.1.82 UNDER REVIEW	******** CRITICAL ********
	DIALUP	AAA999	999 9999				SENT TO FIELD ENG. 26.11.81	STARTED 26.11.81 PROBLEMS	NO LINES IN CITY. NO DATE.
SWANSEA	LEASED	9999/M/YY	Rnnnnnn				STARTED SCHED = 2.12.81	2.12.81	UNDER-GROUND FAULT
	DIALUP	AAA999/999	999 9999			INSTALLED			
MANCHESTER	LEASED	9999/M/YY	Rnnnnnn			29.3.82 ******** CRITICAL ********			

Document Title: EXHIBIT 2 DISCUSSION RECORD	PROJECT HP3000	SECTION 1.1	SHEET 1.1.2
SUBJECT/PURPOSE HP3000 Environmental requirements	PARTICIPANTS LKD C.E		
INTERVIEWER L.K Darnton	LOCATION Phone		DATE dd.mm.yy
START TIME hh:mm STOP TIME hh:mm	FURTHER DISCUSSION REQUIRED ? Y		REFERENCE

DETAILS:

8. AIR CONDITIONING

8.1 In addition to temperature (para 4) and humidity (para 5), it would be wise to provide for air intake from outside room.

8.2 The air intake should provide for dust filtration & would generate a positive air pressure differential between the computer room and the corridor and between the computer room and the car park.

8.3 This positive pressure differential would assist the removal of dust particles.

8.4 Printers should be located near the door, so the positive air pressure differential will assist the transmission of paper & ribbon dust away from the computer room, via the door opening.

8.5 Convector heaters / coolers should not be positioned near printers, because convection currents may be stronger than the currents caused by positive air pressure differential; such convection currents could propel paper and ribbon dust into the computer room, rendering air pressure differential useless as a cleansing medium.

9. Proposals re: ground floor Room 4:

Better than rooms 8/9 by reason of:
- Security
- Layout
- Peripheral access
- Office accommodation

DOCUMENT TITLE: EXHIBIT 3 DISCUSSION RECORD	PROJECT: HP3000	SECTION: 1.1	SHEET: 1.3
SUBJECT/PURPOSE COMPUTER ROOM FLOORING - CABLE DUCTING	PARTICIPANTS: LKD C.E		
INTERVIEWER: L.K. DARNTON	LOCATION: PHONE		DATE: dd.mm.yy
START TIME: hh:mm	STOP TIME: hh:mm	FURTHER DISCUSSION REQUIRED ? yes	REFERENCES

DETAILS:

1. Trunking unlikely to bend under load if metal.
2. Cable = ½" diam, therefore minimum 1"-1½" trunking needed.
3. Trolley may lurch, or push trunking, or both.
4. If trunking pushed, could damage cable, or more likely, terminations.
5. Mains could short, or short with signal lines.
6. If cables defective, or modified to rectify inherent faults, fixed trunking would need to be pulled up - not an engineer function - time delays; same for cable revisions.
7. Personnel tripping over trunking could do more damage than cost of false floor.
8. Trunking not needed if machinery in line, L-shaped or U-shaped. Such layout not possible in room 4 or rooms 8/9.

NOTES RE FALSE FLOORS:

Surface mounted plugs & sockets (flush);
Clearways;
Underfloor fire extinguishers
Repositioning equipment (much easier);
Floor cleaning & polishing easy;

Cable to CPU (Power) = 1" diam; very difficult to conceal safely except in floor cavity;

Thick power cables to disc drives.

DOCUMENT TITLE EXHIBIT 4 DISCUSSION RECORD	PROJECT: HP3000	SECTION: 1.1	SHEET: 4.2.1
SUBJECT/PURPOSE PROPOSED COMPUTER ROOM Initial Environmental Survey	PARTICIPANTS: Mr.K.J. of A.B.C Co. Ltd		
INTERVIEWER: L.K. DARNTON	LOCATION: ON SITE		DATE: dd.mm.yy
START TIME : hh:mm	STOP TIME : hh:mm	FURTHER DISCUSSION REQUIRED ? YES	REFERENCES:

DETAILS:

1. I showed ground plans for room 4, HP3000 specs. We visited room 4 for survey & discussed basic work schedule.
2. FLOOR: Overall height 5"; Ramp slope 1:10 max*;
   Underfloor fire detection est £xxx;
   Total access; flush clean supply sockets.

   *SAFETY REGULATIONS
3. HEAT OUTPUT:

Machine (HP specs)	18,800
Personnel (max 21,000)	9,000
Lighting	3,000
Modems (estimate)	1,500
	32,300 BTU

   TOTAL ESTIMATED = 30,000 - 36,000 BTU
   Plus expansion.
   Plus structural absorption.
4. WALLS + OFFICE PARTITION + DOORS:
   All ¾ hour fire rated; no sliding door;
   Partition may be partially glazed (max 1 sq.m per 4' width); partition right up to structural ceiling (cut slot in false ceiling)

   SEE FIRE OFFICER
5. FIRE: Underfloor detectors; handheld extinguishers probably OK.
   Computer Room classified as Fire Generation/ Power Generation Room; emergency buttons by computer room entrance and office side of partition.

   SEE FIRE OFFICER
6. AIR INTAKE: 1.1/4.2.3
   Inlet/exhaust already exist; blank off old exhaust opening. Install fan to give 500 cfm thru 12" sq duct, with 3-speed control (gives (50 cfm per person normal load x 3) + dust filter resistance + positive pressure differential + peak personnel loading.
   Dust filters = 80% at test dust no.1 and
   95% at test dust no.2 both 12" sq
7. HUMIDITY:
   Air intake inlets humidity, fan blades ensure good moisture mix with air; coolers remove moisture (condense & dump outside). Humidifiers (require plumbing) introduce measured quantity of vapour to reduce static - underworked because of moisture introduced by air intake. 1.1/4.2.5
8. Min 24" below ceiling; more efficient near ceiling; backup unit in case of failure.
9. FLOOR PREPARATION: Heavy wash; hard polyurethane varnish. Move telephone junctions up wall.

Continued 1.1/4.2.2

DOCUMENT TITLE: EXHIBIT 5 DISCUSSION RECORD	PROJECT: HP3000	SECTION: 1.1	SHEET: 5.1
SUBJECT/PURPOSE: PROPOSED COMPUTER ROOM FLOODING RUMOUR	PARTICIPANTS: A.B C.D E.F G.H I.J K.L M.N O.P		
INTERVIEWER: L.K. DARNTON	LOCATION: VARIOUS		DATE: dd.mm.yy
START TIME : hh:mm STOP TIME : hh:mm	FURTHER DISCUSSION REQUIRED ? YES		REFERENCES:

DETAILS:

1. AB in his office:
   Confirmed both rooms 4 & 8 flooded after heavy rain.
2. CD in AB office:
   Flooded through canteen, up passage, into room 4
   Drain overflowed EF/QR
3. EF on phone:
   Blocked drain by bike racks overflowed into works entrance, crept along concrete, didnt flood in true sense of word - thru partition (ie partition pinned to floor - water under & thru partition). Friday night during heavy rain - not noticed until Saturday. Stud walls & carpets soak up water. Problem cured & probably wont happen again, but no guarantee. If drains blocked again, rooms 4 & 8 could flood again. Suggest raised floor installed.
4. IJ in kitchen:
   Flooding occurred outside in drains at bottom of emergency stairs; dont know where it got in. Water rises under fire stairs when drains blocked.
5. GH in factory:
   Water entered via large roller doors (bricked up now), due to flooded drains outside. Concrete ramp now built in front of existing works entrance prevents further flooding. Drains 1.1/5.4.1
   under fire escape couldnt flood kitchen before water evacuates onto car park. 1.1/4.4.2
6. KL & MN:
   During flood one weekend some months ago, standing water was discovered on the MONDAY morning. Weekend maintenance staff had not discovered or suspected flooding in room 8. All lino tiles had to be lifted to expose wet concrete. Boxes in corner Z (see diag) water-logged; all objects & materials on floor had to be lifted up. Corridor was wet, with water seeping through stud walls & skirtings. Very heavy rain.
7. OP in factory:
   Intensity about 1"-2"; damage extensive on ground floor; has happened several times; no heavy rain recently to test whether new preventative measures actually work.

THE SITE SURVEY

This section is a suggested checklist. It will satisfy some requirements but not others. Your project may be bigger or smaller. Use this checklist as a guide to indicate the sort of things you can expect to have to do. If you do not require to perform all listed tasks, omit them. This list is drawn from actual examples of projects undertaken and has been found to constitute a workable method of applying the Guide in a logical sequence.

1. Obtain ground plans for your Company's buildings.

2. Obtain or draw a map of departmental and individual staff locations.

3. Use the Site Planning Workbook materials to create several hypothetical computer rooms to calculate roughly the floor area required to house the hardware. Lay out the equipment in different ways (line, u-shape, l-shape, square, console facing window, etc) and consider how you would feel working in your immaginary rooms.

4. Repeat step 3 for other rooms the DP department will require (Office space, store rooms, paper handling, media storage, etc).

5. Identify from the ground plans, which actual rooms would satisfy the basic space requirements. Select rooms which would provide a DP Department in adjacent or nearby offices. Avoid the DP Manager being located accross the road, or the operator having to wheel trolleys of output through the welding shop or up a small flight of three steps !.

You may find a large room which would house several proposed DP Department rooms if partition were erected. Include these on your list.

6. Using data from step 2, identify staff which would be displaced by the computer if each of your alternative room selections was used. At this point the project must include user departments and higher management on the planning team. This step will require some thrashing out, because no decision can be made until detailed structural surveys have been conducted. However, top level approval must be sought before proceeding any further.

7. When all possible rooms available for conversion have been identified and agreed, use the site planning workbook materials to produce a rough ground plan for each room. See Exhibit 6.

8. Repeat step 3 in greater detail for each room. Spend as much time as you require on this step. Do not hurry or shortcut. A few days spent here will save years of low morale, errors or poor production. Use your imagination to create each room. Play the game of being a fly sized operator walking around the floor of each diagram (see Exhibit 7). Use paper or card cutout of trolleys, people, etc, to create dynamic scale paper models; move the trolleys around your alternative computer rooms to see if you can push them around corners and between equipment.

Do not feel embarrassed to build 'toy rooms' to play with. Build your models and play with them to reinforce your imagination, until, like a child, you can experience, with some degree of subjective reality, actually being in your design.

Visit each room, several times if necessary, to experience its true scale, lighting etc. Visit during different daylight conditions.

Imagine your operator working there every day for hours on end. Consider the operators comfort, reflections on the VDU, changing tapes, walking around equipment to perform various tasks.

Reject rooms which will provide a cramped space. A cramped machine room will created an unhappy operator & errors.

When you have undertaken a subjective journey into each imaginary computer room, begin a detailed survey of each alternative.

9. Refer to the Guide for floor specifications. Examine the floor at each site and consider walls, ceilings, windows. Consider each room from the point of view of physical security. Imagine yourself to be a burglar or sabboteur, attempting to enter from outside or from the corridor. Look at adjoining rooms, including rooms upstairs & downstairs.

Examine the land contours and drainage outside the building; Imagine drains overflowing and spilling into the building. Check the roof construction and condition. Check for electricity supplies, gas or water plumbing running adjacent or through the rooms, perhaps in ceiling, floor or wall voids. Your maintenance department should be able to help.

10. Check the task of running a clean electricity supply into each alternative room. Position the air conditioing equipment in various places and get some initial feedback from your maintenance staff regarding piping runs, mounting condensers or other equipment on the outside of the building (from a structural point of view). Consult higher management about the aesthetics of knocking holes through beautiful facades and bolting electrical equipment on the Georgian masonry !

11. Call Hewlett Packard Customer Engineers. Give them your detailed plans for each proposed computer room and ask for comments about the suitability of conversion of each room. Have your customer engineer visit each room so (s)he can see both the plans and the actual environments at the same time.

12. Consider all feedback obtained from all sources, including users. Older staff will often supply extremely valuable information without which an apparently perfect choice may turn out to be a complete disaster (see Exhibit 5). Select the room(s) you will convert.

13. Return to higher management to discuss the logistics of staff or departmental displacement. Only when this step has been finalised and agreed do you proceed to detailed design work.

14. Using the workbook materials, design a detailed equipment layout for each room you will eventually convert. Approach your design from the point of view of machine requirements and human

comfort. Do not economise on human comfort and health (but be reasonable). Computer environments do produce considerable stress, much of which is not quantifiable by reason of our lack of knowledge.

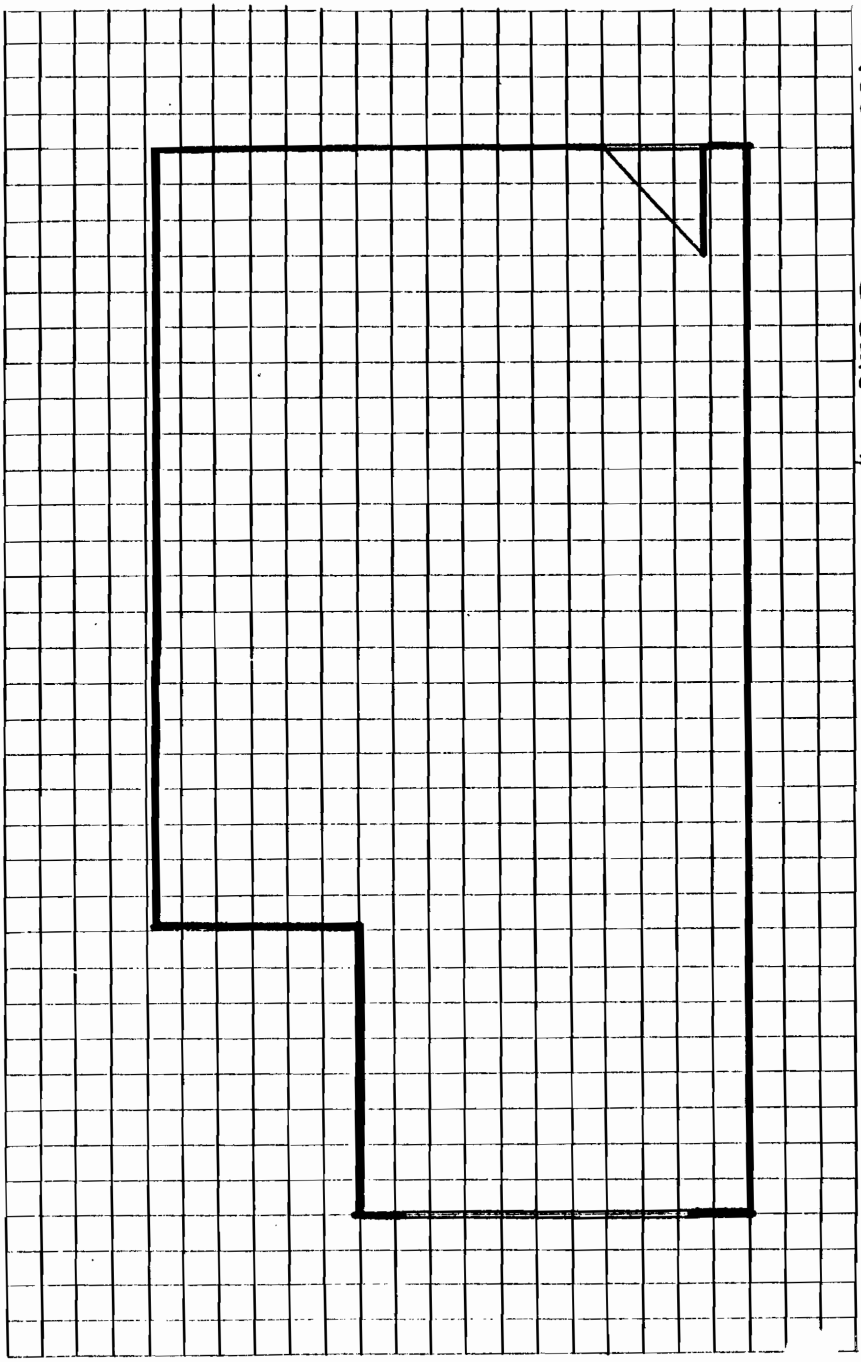
Give sufficient space for operators, engineers and peak personnel loading. Use your operations experience once more to visualise all the normal operator tasks.

15. Produce maps for the floor, heat distribution and floor loading. At this point you have

- i. selected your room(s)
- ii. produced equipment layouts, considering the operator as human being
- iii. produced initial plans from which your Installation Manual will grow.

See Exhibits 7 through 10.

EXHIBIT 6



Room 15 : Scale = ¼":1'

EXHIBIT 7

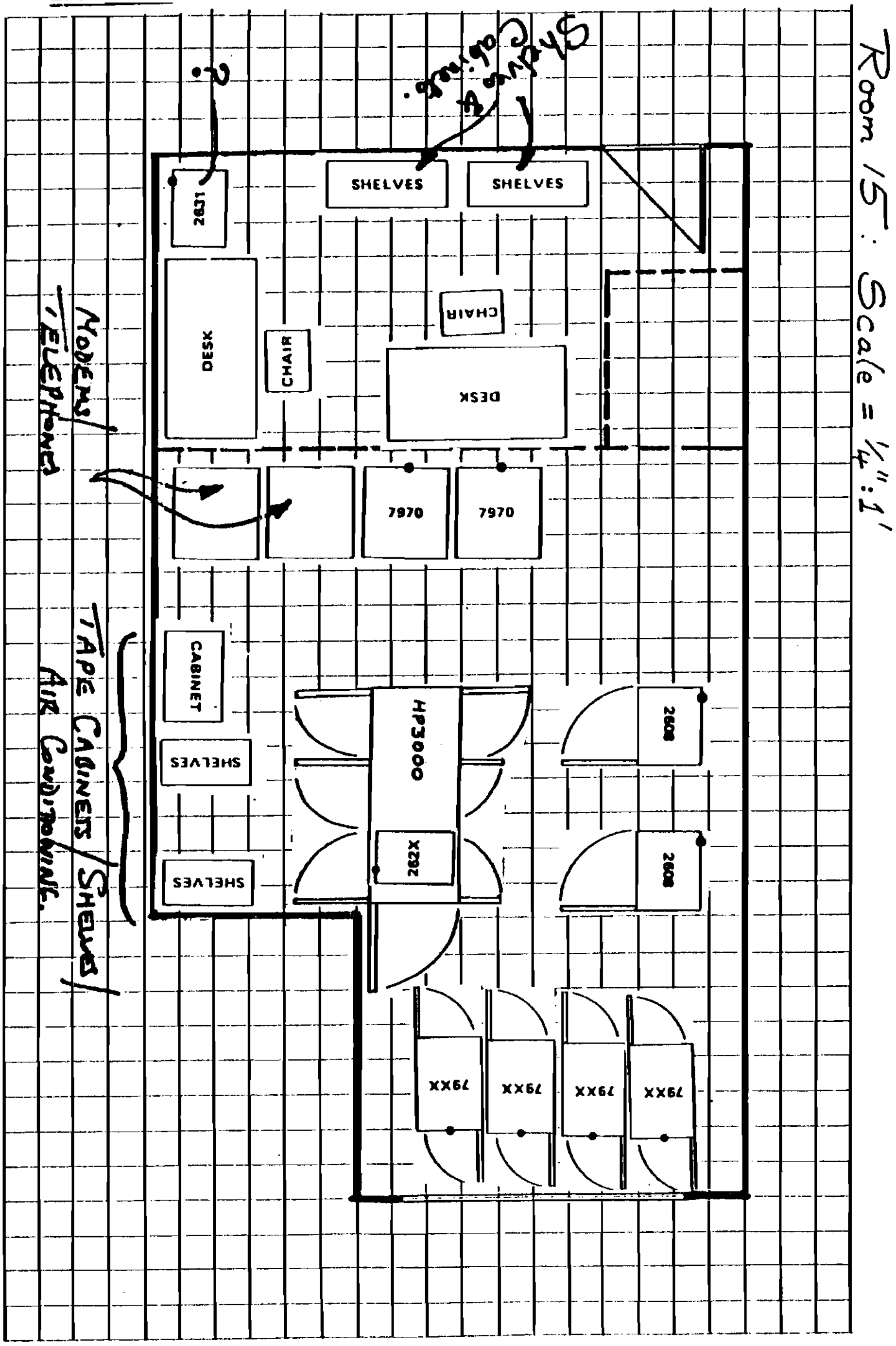

Room 15: Scale = 1/4":1'
Shelves & Cabinets.
?
2631
SHELVES
SHELVES
DESK
CHAIR
CHAIR
DESK
MODEMS/TELEPHONES
7970
7970
TAPE CABINETS/SHELVES/AIR CONDITIONING.
CABINET
SHELVES
SHELVES
HP3000
262X
2608
2608
79XX
79XX
79XX
79XX

EXHIBIT 8

Room 15 : Scale = ¼":1'

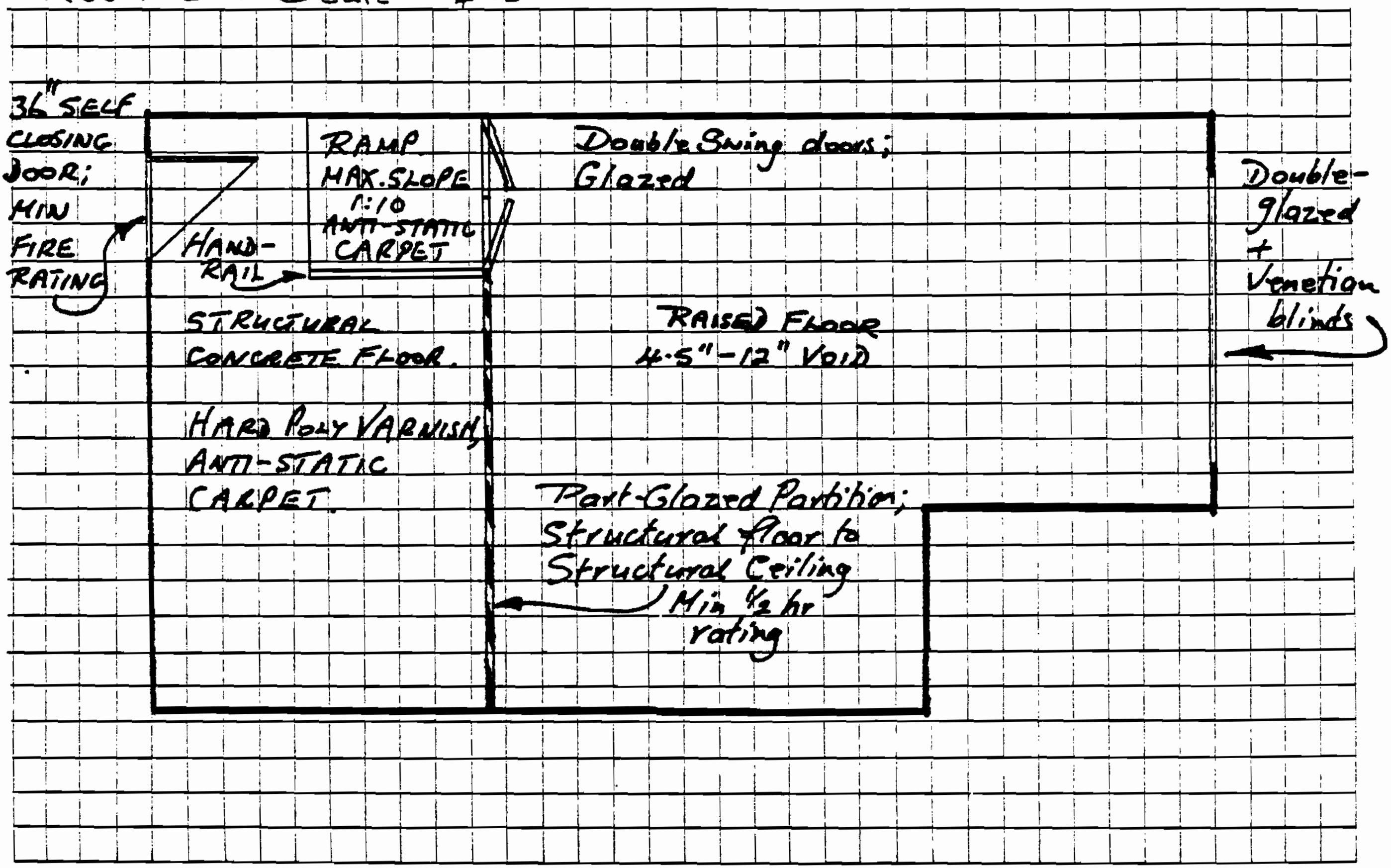
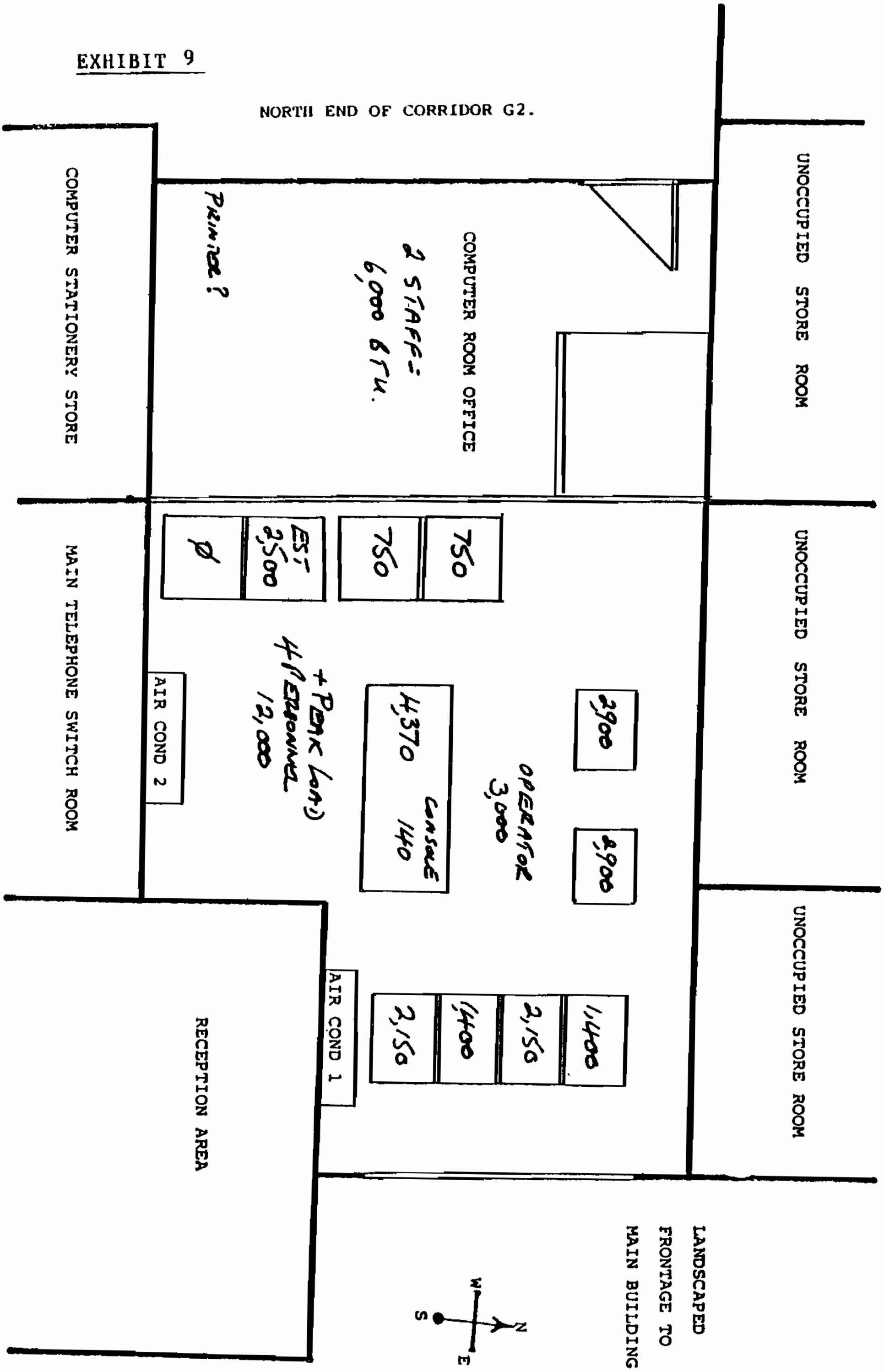
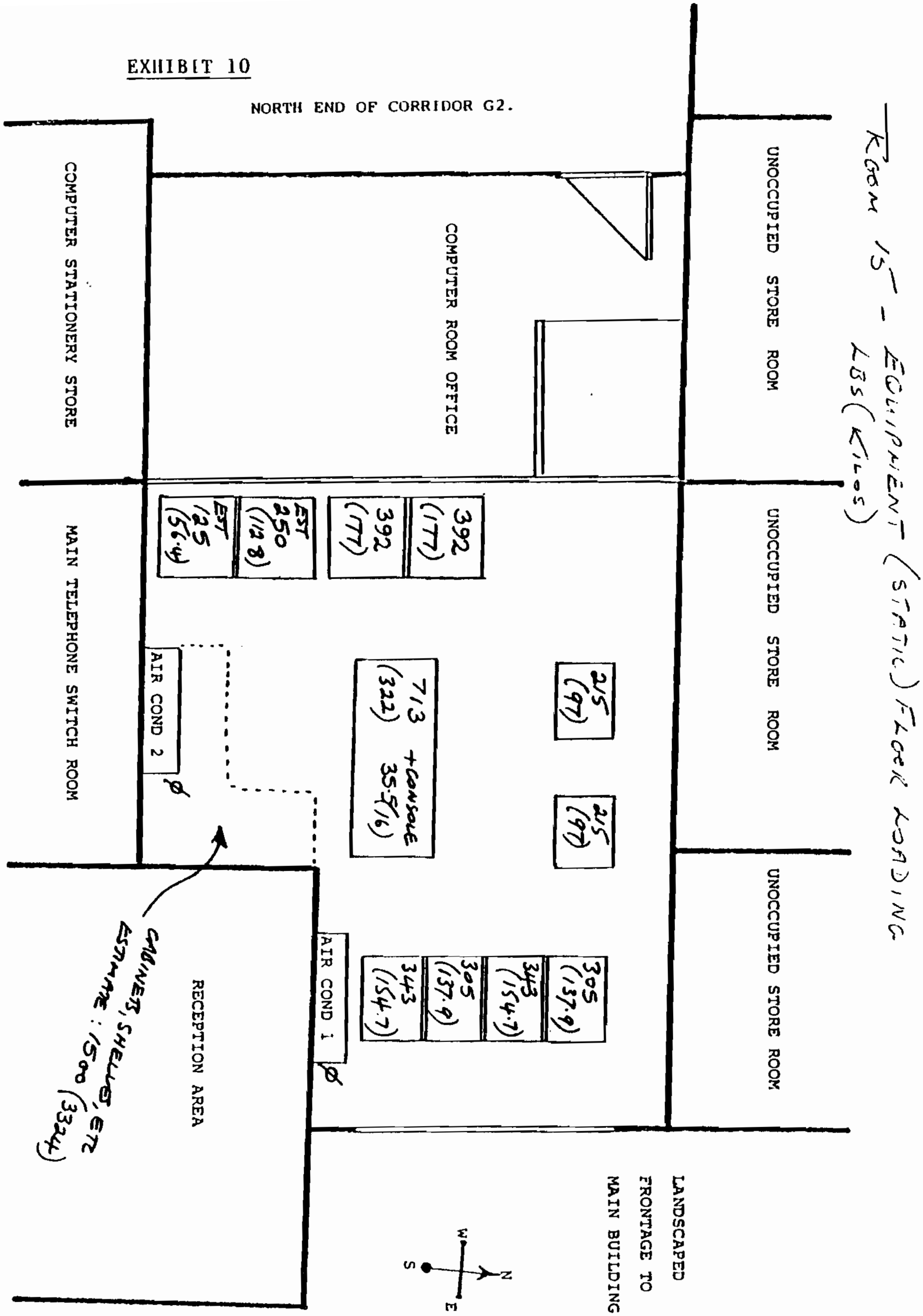

36" SELF CLOSING DOOR; MIN FIRE RATING
HAND-RAIL
RAMP MAX. SLOPE 1:10 ANTI-STATIC CARPET
Double Swing doors; Glazed
Double-glazed + Venetian blinds
STRUCTURAL CONCRETE FLOOR.
RAISED FLOOR 4·5"-12" VOID
HARD POLY VARNISH, ANTI-STATIC CARPET.
Part-Glazed Partition; Structural floor to Structural Ceiling
Min ½ hr rating

EXHIBIT 9


NORTH END OF CORRIDOR G2.
COMPUTER STATIONERY STORE
COMPUTER ROOM OFFICE
PRINTER?
2 STAFF: 6,000 BTU.
UNOCCUPIED STORE ROOM
UNOCCUPIED STORE ROOM
UNOCCUPIED STORE ROOM
MAIN TELEPHONE SWITCH ROOM
Ø
EST 2,500
750
750
AIR COND 2
+ PEAK LOAD 4 PERSONNEL 12,000
4,370
CONSOLE 140
OPERATOR 3,000
2,900
2,900
AIR COND 1
2,150
1,400
2,150
1,400
RECEPTION AREA
LANDSCAPED FRONTAGE TO MAIN BUILDING
W
S
N
E

ROOM 1ST - EQUIPMENT HEAT OUTPUT

EXHIBIT 10


NORTH END OF CORRIDOR G2.
COMPUTER STATIONERY STORE
COMPUTER ROOM OFFICE
UNOCCUPIED STORE ROOM
UNOCCUPIED STORE ROOM
UNOCCUPIED STORE ROOM
MAIN TELEPHONE SWITCH ROOM
EST 125 (56.4)
EST 250 (112.8)
392 (177)
392 (177)
AIR COND 2
Ø
713 (322) +CONSOLE 35.5(16)
215 (97)
215 (97)
AIR COND 1
Ø
343 (154.7)
305 (137.9)
343 (154.7)
305 (137.9)
RECEPTION AREA
LANDSCAPED FRONTAGE TO MAIN BUILDING
W
N
E
S

ROOM 15 - EQUIPMENT (STATIC) FLOOR LOADING LBS (KILOS)

CABINETS, SHELVES, ETC ESTIMATE: 1500 (3324)

16. Using Exhibits 7 through 10 and the plans for the entire building, call in three consultants initially:

i. your local Fire Inspector (call the Fire Brigade for details) and tell him you are installing a computer, have plans of the computer room and would like his advice regarding fire alarms, fire prevention and fire FIGHTING.

Note his comments and advice on discussion records.

ii. your Company's insurers. If they are not experienced in the field of computer insurance, contact a number of Company approved insurance organisations.

iii. Flooring and air-conditioning vendors.
Although you could contract several companies to provide raised floors, partitions, air conditioning, etc, your project will be considerably simplified by contracting a total environment supplier. Much of the more complex scheduling of overlapping tradesmen's work will be handled by a single supplier.

Check that the suppliers are familiar with all local BUILDING, FIRE AND SAFETY REGULATIONS IN FORCE, and ensure that compliance with all such regulations will be built into any purchase orders.

17. Take note of any recommendations or demands made by the Fire Inspector and insurance company. Modify plans if necessary.

A practical example is one company where an excellent computer room was designed with very carefully thought out office accommodation for DP staff. Original occupants of the rooms had been relocated and the project was in full swing. The fire inspector visited the site and reviewed proposed plans. The plans were rejected because the only emergency exit for DP staff would have been through the computer room itself; this being a fire/power generation area, could not be used as a fire escape route. The Inspector demanded that a fire escape door be cut into an external masonry wall leading directly to the outside.

Higher management would not permit such a door to be cut into the front facade of the building for various reasons. The entire computer room design was scrapped and restarted from square one.

In this case several days design work was lost; however, the consequences of not consulting the Fire Inspector first would have been closure of the DP Department and cessation of business until requisite fire exit had been constructed. This event may not have occurred until several weeks or months after going live !

18. Another example of interaction with the Fire Department is:

The Fire Inspector demanded that one Company install automatic underfloor halon fire extinguishers, to be triggered by the fire alarm system. However, because of the structural detail of the room, a short ramp up to a raised floor limited the floor height to 5" overall (too small a floor void to accommodate the recommended type of fire equipment). Discussion with the Fire Inspector and

Insurers led finally to an agreement to install manual halon fire extinguishers, two strategically positioned emergency off buttons (which would shut down the entire electrical supply and air conditioning plant), and a manual break-glass unit in a corridor outside the computer room itself. In addition, a more comprehensive alarm system was required, with alarms in four locations in the building.

In this case, if the computer electrical circuits had been installed before discussions with the Fire Inspectors, a considerable amount of rework would be necessary, requiring the complete closure of the computer system.

19. In another example, a three-phase supply was to be run into the computer room, each phase to be used to supply different circuits within the same room. It was the Fire Inspector who pointed out the dangers of having equipment on different phases too close together, or of an engineer plugging test equipment into a socket suplied by one phase, to test machinery supplied by another phase.

The dangers of this arrangement were verified by Hewlett Packard engineers.

20. Only when all plans have been finalised and agreed with the Fire Inspector, Insurers and any other authorities involved, should you schedule actual construction work.

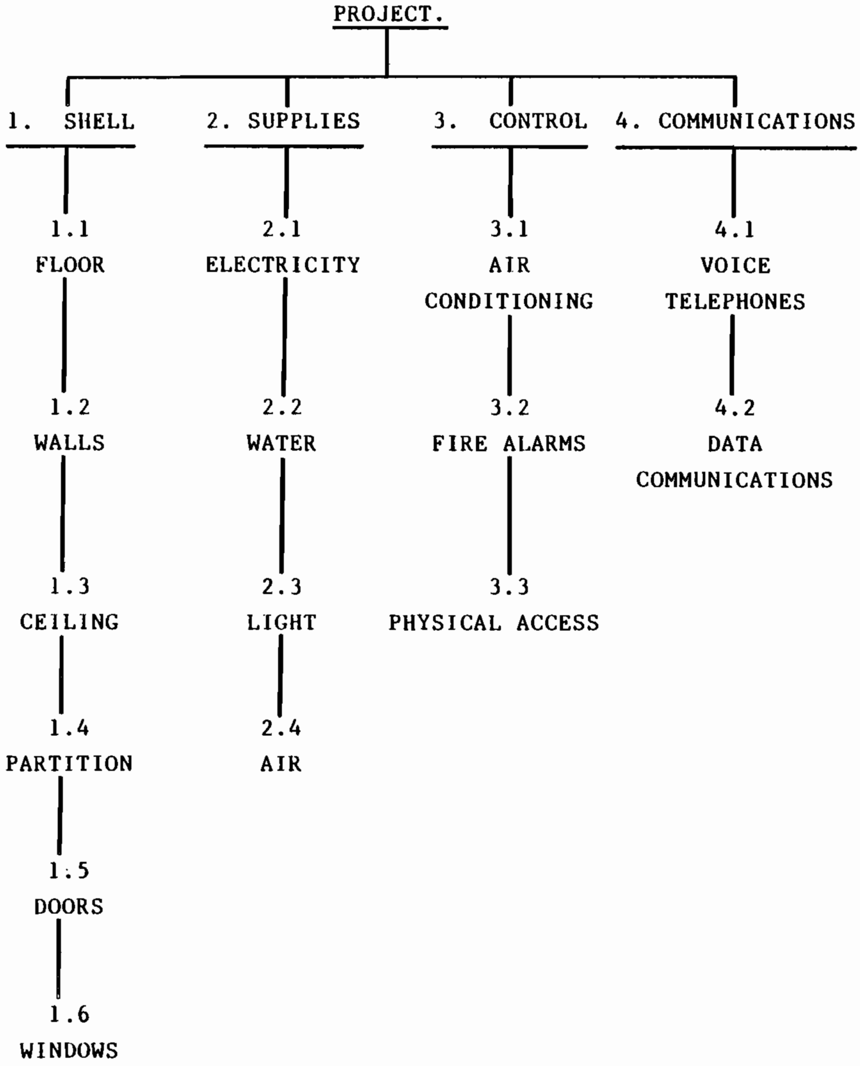
Plans such as Exhibit 7 can be used to draw electrical circuits, air conditioning supplies,fire alarm circuits, communications circuits; floor tile cutouts will be required for power and data cables; these can be detailed on a blank overlay derived from Exhibit 8.

21. The entire project can be broken down into constituent parts (Exhibit 11) such as the SHELL, SUPPLIES, CONTROL, and COMMUNICATIONS. This represents the concept of the shell or enclosure containing the computer environment, the energy supplies required to feed the environment, the controls required to maintain the environmental condition and finally, the communications between the environment and the outside world. Other different conceptual divisions can be formulated to suit your own particular project requirements or scope of responsibility.

Each 'leaf' of the 'tree' (Exhibit 11) is expanded (Exhibit 12) into an unordered list of tasks to be performed. The list of tasks for each 'leaf' is drawn from your file of discussion notes, the Guide and your plans, diagrams, etc.

Each unordered list is, by a logical process somewhat akin to writing a program and thoroughly deskchecking it, rearranged into an actual work schedule. This schedule can then be used as the basis for the creation of your network (Fig 2) and finally your bar chart (Fig 1). Progress charts (Exhibit 1) can be drawn up where necessary - usually where a detailed level of the project contains many detailed components which require critical control, for example the installation of several private circuits with associated dial back-up lines, modems, etc; components such as these involve many remote activities which can usually be monitored only by telephone; thus frequent communications are required, with very accurate discussion records.

DOCUMENT TITLE: EXHIBIT 11 SUMMARY WORK SCHEDULE - COMPUTER ROOM	PROJECT: HP3000	SECTION: 1.0	SHEET: 1.0


PROJECT.
1. SHELL
1.1 FLOOR
1.2 WALLS
1.3 CEILING
1.4 PARTITION
1.5 DOORS
1.6 WINDOWS
2. SUPPLIES
2.1 ELECTRICITY
2.2 WATER
2.3 LIGHT
2.4 AIR
3. CONTROL
3.1 AIR CONDITIONING
3.2 FIRE ALARMS
3.3 PHYSICAL ACCESS
4. COMMUNICATIONS
4.1 VOICE TELEPHONES
4.2 DATA COMMUNICATIONS

DOCUMENT TITLE: EXHIBIT 12 SUMMARY WORK SCHEDULE - COMPUTER ROOM	PROJECT: HP3000	SECTION: 1.0	SHEET: 1.1

LEVEL 1.1 - FLOOR

TASK No	TASK	DATE COMPLETE	COMMENTS
1.1.1	Vacate room		
1.1.2	Lift carpet		
1.1.3	Sweep/vacuum floor		
1.1.4	Anti-static floor cleanse		
1.1.5	Hard polyurethane varnish		
1.1.6	Lay raised floor		
1.1.7	Fix skirting trim		
1.1.8	Fix handrail housings		
1.1.9	Prepare/fix handrail		
1.1.10	Cut housings for flush mains sockets		
1.1.11	Fit flush mains sockets		
1.1.12	Cut entry hole for mains cables under main distribution panels		
1.1.13	Carpet in office		
1.1.14	Carpet in doorwell		
1.1.15	Carpet on ramp		
1.1.16	Skirting trim in office		
1.1.17	Cut signal cable entry points to peripheral cabinets		
1.1.18	Cut special access for communications cabinet base		
1.1.19	Final cleanse before computer delivery		

22. Purchase Orders for major items.

When vendors visit your site, you will notice two types. One will give you some glossy pamphlets, a list of advantages of their products and ask you to select the model you want.

The other type is more interested in a long term relationship. He will conduct a thorough survey.

For example, let us examine air conditioning. The first type may look at your heat output calculations and show you a number of units which will satisfy your calculated cooling requirements. The second type will request a step ladder so he can inspect the existing ceiling void; he will pull the carpet up to see the floor, want to know about the material construction of the walls, examine the exterior walls for structure and condition, examine adjoining rooms (including above and below); he should consider the local environment, peak personnel loading, etc, etc. You will know that you have had a thorough structural & environmental survey. His facts & calculations which you have recorded on your discussion records, can be checked against calculations using Hewlett Packard's formulae in the Guide. He will provide you with a list of satisfied customers.

I have always followed up references provided by potential suppliers. You would be surprised how some telephone calls have resulted in ".....well, we took our first three installations from XYZ Co.Ltd., but we found their service to be so bad, we have now changed to PQR Ltd; PQR have now installed 25 complete environments to our satisfaction and their service is excellent.". Why XYZ Co. Ltd., still provides references such as these I dont know. I suspect that very few people actually check the impressive lists of clients which these suppliers give out. ALWAYS ask for a fairly long list of references and ALWAYS check them.

When writing a purchase order, do not simply order the goods from the catalogue. Make the order subject to various previos agreements. For instance instead of ordering:

1 Air conditioner unit type AD5034

order instead

1 Air conditioner unit type AD5034 as proposed by PQR Ltd (Mr. XXX) on dd/mm/yy to satisfy the environmental requirements discussed and agreed between our Mr. Darnton and your Mr. XXX on dd/mm/yy and detailed on the Schedule to this purchase order attatched hereto.

The Schedule referred to in the purchase order should list the environmental controls required and the equipment to be installed now and in the future, and normal and peak staff loads. It should state that PQR Ltd has conducted a survey and that if anything in the schedule is found by PQR Ltd to be incorrect, the co-ordinator should be contacted in order that an amended purchase order schedule can be drawn up, and a new purchase order delivered.
PQR's delivery and installation of the ordered items constitutes their agreement with everything which is written into the purchase order schedule.

I have, on several occasions, relied upon such schedules, in order to receive timely or free re-work in the case of design faults which were the sole responsibility of the suppliers.

Ideally, all your supplier relationships would be trust-based, but this rarely works out in practice. Some suppliers are proud of their name and products and will give excellent service to maintain their goodwill. Others need to be bound by law. My reasons for tying suppliers down are that in the event of disagreements between the supplier and my Company, we can fall back on a strictly legal contract. Both the supplier and I are protected. Another point of view is that a supplier's salesman and I may have hit off a good relationship, whereas his or my successor may be very different persons; they will not have the knowledge of the system which I possess; even where documentation is perfect, successors are not the creators of the system; most people will protect their own creations and there is a deep rooted instinct to replace those of others - perhaps a hangover from "marking territory" ?

23. SELF DISCIPLINE AND PROJECT CONTROL

From here on, the key to your smooth operation is efficient documentation and immediate follow-ups to problems, with the most urgent attention paid to tasks falling on the critical path. Regular reviews should be executed. Where these are reliant upon information from external sources, you should exercise a default silence period; that is, if you hear nothing for n days, make a status enquiry.

If you are not disciplined, this will be reflected in your project status at any moment in time. Unfortunately, delays often tend to behave cumulatively, such that a short period of inefficiency is very difficult to rectify later without pumping additional resources into the project.

You will learn very quickly that time is a resource with its own momentum. Even if you stand still, time will continue. Time also consumes money; the two are inseparable. You will also discover that however well you plan your schedules taking into account all known and measurable factors, there will always be some delay, error, accident, etc., which will pull you back off schedule. It is absolutely imperative to build contingency time into your milestone estimates. Various attempts to define scientifically how much contingency to build in, have been made; however it appears that each person in a given situation will require his or her own average contingency time. This can be determined by measuring your time estimates for project components from the outset; measure actual time taken and calculate your variance. The variance will relate generally speaking to an individual's level of skill; however, various conditions may interfere with an individuals performance and this interference will fluctuate from day to day.

Building in sensible, realistic contingency time is the project planning & control analog of safety tolerances in mechanical or electrical engineering. In other words, build in no safety margin and your system (in this case your designers and construction staff) will fail under the constant stress of full load and the intermittent stress of excessive load.

Therefore, omission of contingency time will often extend the project.

FURTHER READING

1. HP3000 Site Planning and Preparation Guide.

2. HP3000 Site PlanningWorkbook.

3. Recommendations for The Protection of Computer Installations Against Fire.

   Fire Offices' Committee and Fire Offices' Committe of Ireland
   Aldermary House, Queen Street, London, EC4N 1TT,
   December 1979.

4. Fire Precautions Act 1971
   HMSO

5. Introduction to Network Analysis:

   5.1 Systems Analysis for Business Data Processing
   H D Clifton
   Business Books Ltd., 24 Highbury Crescent,
   London N5

   ISBN 0 220 66369 6

   5.2 Operational Research
   W.M. Harper
   M&E Handbooks
   MacDonald & Evans Ltd
   Estover, Plymouth, PL6 7PZ.

   ISBN 0 7121 1514 5

These books provide an introductory level text which should be sufficient to enable most site planners use network charting and analysis quickly and to advantage.

Making Utilities Non-utilitarian

Presenter

Chris Shinn

Louis Dreyfus & Co. Ltd.

Abstract

There exists a wide range of system utilities available from HP and other vendors to enable rapid development of reports and to meet ad hoc user requirements. The problem for commercial users is how to make them usable and acceptable to the non-technical user.

Several years ago we developed a "user friendly" job initiation utility whereby parameters could be substituted into a batch stream file from an interactive session. This program which we have called FIJI (Friendly Interactive Job Interface) has evolved into a versatile job control language which we use to "front end" system tools and reporting software (ASK,RPG etc.).

The presentation will examine the design of the user interface and demonstrate how easily interactive dialogues can be constructed. It is intended for system managers and programmers interested in using cost effective software aids without the necessity for their users to acquire special skills. The benefits of this approach will be illustrated with examples taken from the several hundred offline facilities we use in our shipping and trading systems.

Computer Museum

USING UTILITIES IN COMMERCIAL APPLICATIONS

In many installations systems tools are frequently used to aid development but rarely form part of the final application itself. Design considerations concerned with providing good response times and optimum resource utilisation from data entry and enquiry transactions often outweigh arguments in favour of employing utility software. However, this is not the case with the offline components such as reporting, periodic re-organisation and archiving procedures.

For example, we know that to create an IMAGE key for every possible report selection path (or even worse, use sorted chains) will simply reduce overall performance. On the other hand, the use of a fast extract utility, sort and print, not only cuts report development time but also reduces the need for database design compromises. It can also be faster in execution e.g. where NOBUFF I/O is used.

The scope for using utilities promises significant cost savings. Offline procedures in our installation represent about half of our software development workload. This includes maintenance enhancements which largely comprise further reports and analyses to obtain maximum benefit from the information base that has been built up.

THE NEED FOR AN INTERFACE

It might be tempting to place utilities directly into the hands of our users. There are, however, several good reasons for not adopting this approach.

In the first place general purpose software by its nature must be "all things to all men". This necessitates the use of abstract computer terms such as field description, dataset names etc., rather than terminology familiar to the user's own discipline (account designation, customer reference etc.).

Secondly, users may view utilities as solutions in themselves. As familiarity is gained with a tool there will be a tendency to "use the spanner as a hammer" rather than call on the services of a trained analyst to determine the appropriate solution to their problem.

Thirdly, not all tools are made to the same standard and quality - they often have operational quirks, limited data protection and poor diagnostics/help facilities.

In summary, they can provide much rope by which users can hang themselves.

DESIGN CHOSEN

To overcome these problems, we decided to employ an interface between the user and utility software. The proposed method was for an interpreter that would process a command type file to establish user dialogues and provide control logic. Following consideration of the possible uses for utilities (i.e. how best to handle offline requirements) it was decided to make the interface a batch job interface. This would avoid tying up the user's terminal when offline tasks are required (mainly reporting) and prevent high volume processing from degrading online transactions.

A further technical consideration at the time was the difficulty of interactively executing a command type file with imbedded program data for $STDIN files. Although the arrival of MPE IV and the ability to re-direct $STDIN input has removed this problem, we have not felt the need to extend the interface to offer online utility processing via process handling.

The interface program we now use, named FIJI, processes a job command file previously created using an Editor (we use QEDIT work files for speed). The command file is similar to a normal job file but contains imbedded parameters and special commands. When a parameter is encountered, the user is prompted with the parameter text. His/her response is used to replace the parameter (and any subsequent occurrence) in the output stream file. The special commands determine logic control and provide for manipulation of help/display text.

Schematically:

```
                             Command file
                                  !
                                  !
                                  V
  Online dialogue ----> FIJI
  and entry                       !
                                  !
                                  V
                               Job file
                                  !
                                  !
                                  V
                           MPE Job Stream
```

The output file created for MPE processing is normally automatically streamed into the batch queue. Alternatively, jobs that require scheduling overnight can be transferred to our scheduler. We intend to incorporate a linkage mechanism this year, but since the majority of our overnight jobs are fully automatic (requiring no run-time parameters) this has been assigned a low priority.

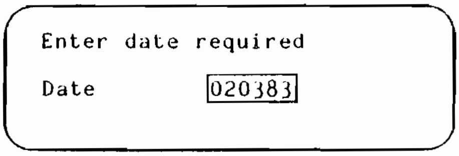
PARAMETER SUBSTITUTION METHOD

As stated above, the command file contains imbedded parameters. These are used to prompt the user for a response which is substituted into the output job stream. This simple example illustrates the basic form:

Input Text:

```
:JOB DAYBOOK,MGR.ACCOUNTS
:RUN PDAYBOOK
!DISPLAY Enter date required
%Date(6)
:EOJ
```

Online Prompt:



```
Enter date required

Date          020383
```

Output Job:

```
:JOB DAYBOOK,MGR.ACCOUNTS
:RUN PDAYBOOK
020383
:EOJ
```

The "%" character is used to identify that a parameter is to be inserted in this position. The text following the "%" (which can include spaces, quotes or any other characters) is displayed as a prompt to the user. The number that follows in round brackets signifies the length in characters allowed for the response and terminates the prompt text.

DOUBLE SUBSTITUTION

In addition to direct substitution, the user's response can be used to select imbedded values. This allows the interface designer to base prompts/replies on what is best for the user rather than what the utility syntax dictates.

For instance, a Query FIND might be based on a month range for producing a quarterly report. The following command lines would only require the quarter to be entered:

```
!DISPLAY Please enter quarter required (1-4)
FIND DB-MTH IB %Quarter? (1)[1=01,03][2=04,06][3=07,09][4=10,12]
```

The value enclosed in square brackets (left of equals sign) is compared with the user's reply. Where the same, the imbedded text (right of equals sign) is used to perform the substitution.

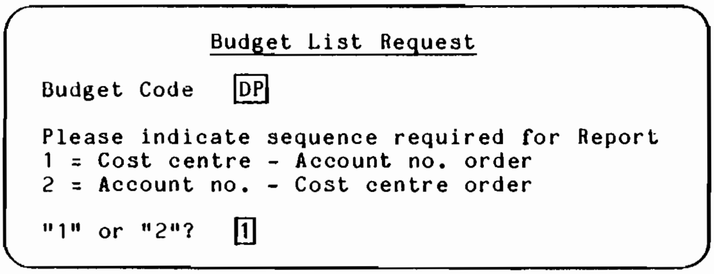
SUBSTITUTION EXAMPLE

A Budget List is required with selection by Budget Code and the option to choose the sequence by which details are reported. Budgets aré held within an IMAGE dataset with the Account and Cost centre codes as the only Search items. The report job uses a fast copy utility, TRANSF, to extract budgets for the Budget Code selected. These records are then sorted according to the order requested prior to input to a print program.

Command Text:

```
!DISPLAYC D,Budget List Request
:JOB BUDGLIST,MGR.ACCOUNTS
:RUN TRANSF.PUB.SYS
FROM=D:LDACTS.PUB/PASS,5,BD-BUDG TO $NEWPASS AND
B111="%Budget Code(2)"
EXIT
:FILE INPUT=$OLDPASS
:FILE OUTPUT=$NEWPASS
:RUN SORT.PUB.SYS
!DISPLAY Please indicate sequence required for Report
!DISPLAY 1 = Cost centre - Account no. order
!DISPLAY 2 = Account no. - Cost centre order
KEY 1,3; 112,2
KEY %"1" or "2"?(1)[1= 4,4; 8,10 ][2= 8,10; 4,4]
END
:FILE SORTBUDG=$OLDPASS
:RUN PBUDGET.PROGRAMS
:EOJ
```

Online Dialogue:



```
                    Budget List Request

Budget Code    DP

Please indicate sequence required for Report
1 = Cost centre - Account no. order
2 = Account no. - Cost centre order

"1" or "2"?    1
```

Output Job:

```
:JOB BUDGLIST,MGR.ACCOUNTS
:RUN TRANSF.PUB.SYS
FROM=D:LDACTS.PUB/PASS,5,BD-BUDG TO $NEWPASS AND
B111="DP"
EXIT
:FILE INPUT=$OLDPASS
:FILE OUTPUT=$NEWPASS
:RUN SORT.PUB.SYS
KEY 1,3; 112,2
KEY  4,4; 8,10
END
:FILE SORTBUDG=$OLDPASS
:RUN PBUDGET.PROGRAMS
:EOJ
```

PROGRAMMATIC CONTROL

As well as substituting text, special commands can also be used to offer programmatic control over the output job. It is obviously important that the use of the interface control language does not become a time consuming development task in itself. With this in mind, certain restrictions are built into the design. First, all prompted entry is held internally as unique parameters which cannot be overwritten. This provides for the ability to re-prompt the user for one or more parameters with automatic "restart". Second, all logic control is achieved through the use of simple GOTO or DO commands. Jobs are prevented from indefinitely looping by only allowing GOTO forwards and limiting DO executions to a specified number of times.

The example below is a simple Accounts List report request that allows optional selection by company. Note the simplicity of the user dialogue in contrast to the syntax required by Query. The example also serves to illustrate the GOTO command and shows the use of the ERROR command to re-prompt user entry.

Command
Text:

```
!DISPLAYC D,Accounts List Request
:JOB ACLIST,MGR.ACCOUNTS
:RUN QUERY.PUB.SYS
XEQ ACDEF
!DISPLAY Do you wish to report on ALL companies?
%"Y" or "N"(1)[Y][N][ELSE=!ERROR Please enter "Y" or "N"]
%"Y" or "N"(1)[N=!GOTO COMPANY SELECT]
F ALL GL-ACCODE
!GOTO CONTINUE
%%COMPANY SELECT
!DISPLAY Enter company code required
F GL-ACCODE IB %Company(3)0000000,%Company(3)9999999
%%CONTINUE
XEQ ACLIST
EXIT
:EOJ
```

Online
Dialogue:

```
           Accounts List Request

Do you wish to report on ALL companies?

"Y" or "N" Z

Please enter "Y" or "N"

"Y" or "N" Y
```

Output
Job:

```
:JOB ACLIST,MGR.ACCOUNTS
:RUN QUERY.PUB.SYS
XEQ ACDEF
F ALL GL-ACCODE
XEQ ACLIST
EXIT
:EOJ
```

ONLINE OPERATION

All screen handling is carried out in character mode. This was chosen in preference to a forms-fill approach for simplicity and flexibilty. It is simple in that the embedded parameters in the command file can be used as prompts, obviating the need for forms files and mapping descriptions. It is flexible since prompts can be conditioned by previous responses. This avoids the confusion that can arise from using screen forms containing mutually exclusive selections i.e. complex error messages and fields taking on different meanings depending on other values entered.

The interface can be run as a stand-alone program, from a UDC or via a menu selection program using process handling. The simple :RUN method is normally only used for internal jobs since the user must supply the command file name.

Running from a UDC or menu, however, allows the user's job choice to be pre-set by using file equations to inform the interface which command file has been selected. The screen is then cleared prior to display. Blocks of display text and entry lines are automatically separated by blank lines to maintain an uncluttered appearance. Prompts for user entry are followed by an inverse-video box set to the length of the parameter required. The user can type in EXIT at any time to abort a job or type HELP to call up a block of pre-defined help text.

To further ease operation, replies can be entered using either the ENTER or RETURN key. Users find this helpful where other online transactions, such as data entry, use block mode. The technique involves preceding the prompt with a special character (we use an underline) which has been defined as a line-termination character via the FCONTROL intrinsic (control code 25). Function keys can also be used for HELP and EXIT requests (the escape sequences are simply translated on input).

HELP FACILITY

A block of help text is specified by use of a HELPMSG command. In common with other display commands, text can be output with display enhancement. In the example below, the character 'J' preceding the message will cause the help text to be shown in half-bright inverse-video on the user's terminal.

```
!HELPMSG J, The report shows all transactions for this year
!HELPMSG J, up to and including the MONTH entered.
!HELPMSG J, Enter the month as a numeric (1-12).
!HELPMSG J, Note that months beyond the current Posting
!HELPMSG J, month are invalid.
```

Several blocks of help text can be designated within a job. The last block preceding the parameter currently prompted for will be selected for display. Thus, the help given can be general or specific to each prompt offering a step by step guide.

VALIDATION FACILITIES

The validation options available aim to provide a standard of user friendliness comparable to our other inhouse programs.

They are:

- Format checks with editing

- Range checking

- Validation against data already held, typically system codes

FORMAT CHECKING/EDITING

Parameters can be checked for format using a VALIDATE command. Editing facilities allow flexibility regarding user entry. For example, where a numeric reference is required,

```
!VALIDATE Voucher Ref(10),RJ,ZF,C,N
```

will right-justify and zero-fill the user's entry as well as validating that it is non-zero (compulsory) and numeric. This avoids the type of confusing situation that can arise when a selected reference entered as "123456" is not correctly interpreted as "0000123456".

DATE MANIPULATION

Dates can also be validated and edited into a form compatible with the way in which they are held on file. Typically, this enables extraction of records on the basis of an effective date or date range. For example, dates entered in DDMMYY (European) format can be reversed into YYMMDD form for field comparison using a DEFINE command as illustrated below:

```
!VALIDATE Start Date(6),D,C
!DEFINE S-DAY(2)S-MTH(2)S-YR(2) = Start Date(6)
:RUN ASK.PUB.SYS
XEQ DEFHIST
F HIST-DATE GE %S-YR(2)%S-MTH(2)%S-DAY(2)
XEQ HISTREP
EXIT
:
:
```

RANGE CHECKING

The use of the ERROR command in double substitutions provides a method method to check user entry against pre-defined values as shown here:

```
!VALIDATE %Month(2),RJ,ZF,N,C
%Month(2)[LT:01=!ERROR Invalid Month!]
%Month(2)[GT:12=!ERROR Invalid Month!]
```

CROSS FIELD VALIDATION

Replies can also be validated against each another. The ERROR command allows for re-prompting of groups of mutually dependent parameters as demonstrated here:

Command
Text:

```
          :
   :RUN TRANSF.PUB.SYS
   !DISPLAY Enter range of transaction references
   FROM D:HIST/PASS,5,TRDET TO $NEWPASS AND
   BO >= "%First Ref(8) AND BO <= "%Last Ref(8)"
   %Last Ref(8)[LT:%First Ref(8)=!ERROR 2,Last before First Ref!]
   EXIT
          :
```

Online
Dialogue:

```
   Enter range of transaction references

   First Ref          ABS03050

   Last Ref           ABN90577

   Last before First Ref!

   First Ref          ABN90577

   Last Ref           ABS03050
```

Output
Job:

```
          :
   :RUN TRANSF.PUB.SYS
   FROM D:HIST/PASS,5,TRDET TO $NEWPASS AND
   BO >= "ABN90577" AND BO <= "ABS03050"
   EXIT
          :
```

REFERENCE TO EXTERNAL FILES

No validation facility would be complete without the ability to reference external files. Almost all applications will use codes for companies, products etc. File look-up enables validation of the codes and relationships between them e.g. is a Supplier code applicable to a Company selected?

Files are declared using a form such as:

```
!FL1 VENDMAST.LIVE.ACCT                (for serial or KSAM files)
!FL2 DBPDS/PASS,5,CERTDET.CERTCODE     (for IMAGE files)
```

Records can be accessed by key value, record number or sequentially. For example:

```
!IN1 KEY=%Vendor Code(8)
!IN2 NEXT
```

A special internal parameter, 'STATUS(2)', indicates whether the read is successful:

OK = Successful
NF = Not found (using key or record number)
EN = End of file (or end of chain for IMAGE detail sets)

ACCESSING RECORD DATA

Record data can be accessed by use of a DEFINE command for substitution of 'internal' parameters within the command file.

```
!FL5 CODEMAST.PUB
   :
!IN5 KEY=CO-%Company(3)
%STATUS(2)[NF=!ERROR Company does not exist!]
!DEFINE .(6)COMPNAME(20) = RECORD(*)
!DISPLAY Company: %COMPNAME(20)
   :
   :
```

A company code of, say, 234 would be used to form the key "CO-234" to access the file CODEMAST. The DEFINE places the company name in an internal parameter COMPNAME which is then displayed as confirmation of the company selected. Note the use of a "filler" field (full stop followed by length) to skip over the key which is the first part of the record itself.

ANOTHER VALIDATION EXAMPLE

The Service History request job below shows how FIJI commands can provide as much information and assistance as would be expected from any other online transaction. Seafarers are selected by a unique reference called the Discharge Book number. This example shows the portion of the job concerning its validation. The reference is checked for format, validity within the system and whether it is currently included in the Service History held. Help text reminds the user how to find a seafarer's Discharge Book number where it is not known.

Command Text:

```
!DISPLAYC D,Service History Report Request
!HELPMSG The Seafarer Discharge Book Number must be entered.
!HELPMSG Use the Name Index via Main Details Display where
!HELPMSG this is not known.
!FL2 DBSVC.PUB/MAINT,5,SVCHIST.DBNO
      :
!VALIDATE Dis. Book No.(10),X,C,LJ
!IN2 KEY=%Dis. Book No.(10)
%STATUS(2)[NF=!ERROR Non-existent Seafarer!]
%STATUS(2)[EN=!ERROR No Service Record for this seafarer]
      :
```

Online Dialogue:

```
              Service History Report Request

  Dis. Book No.  [R908345   ]

  Non-existent Seafarer!

  Dis. Book No.  [R903845   ]

  No Service Record for this seafarer

  Dis. Book No.  [HELP      ]

  The Seafarer Discharge Book Number must be entered.
  Use the Name Index via Main Details Display where
  this is not known.

  Dis. Book No.  [EXIT      ]
```

The first code entered, R908345, could not be found in the IMAGE master set linked to the detail set specified. A value of 'NF' was thus returned in the STATUS parameter. The second code, R903845, exists in the master file but not in the detail set. The STATUS returned 'EN' indicating end of chain. Following entry of "HELP", the block of HELPMSG text was displayed. In the example the user elects to abort at this point via EXIT.

A MODULAR APPROACH TO REPORTING WITH RPG

Many FIJI jobs within our installation comprise an extract, sort and print using RPG. Although RPG is not strictly a utility, it does share some general characteristics. Firstly, it has excellent high-level report formatting and control capabilities and can therefore be classed as a rapid development tool. Secondly, it suffers from the same shortcomings regarding online interaction. Although it is not impossible to develop a reasonable user interface it is very time consuming. Indeed, where its use is stretched beyond reading fixed format files and outputting reports it becomes rather unwieldy and loses its cost effectiveness.

However, used in conjunction with an interface, report modules need only be simple. The interface takes care of online interaction and is able to translate selection requirements for the extract and sort utililtes through manipulation of their parameters. This modular approach allows independent development/modification of the selection and report production components.

The Daily Cash Book request job listed below illustrates the technique. The job prompts the user for a date range (Start and End dates). The dates are converted to YYMMDD format for field comparison by the fast copy utility, TRANSF. Journal entries for cash items are identified by a non-blank Cash Code. Extracted records are then sorted prior to input to the RPG print program.

```
!DISPLAYC D,Daily Cashbook Journal
!OPTION U
:JOB CASHBK,%USER(8).CMDT,ACCOUNTS
!HELPMSG Reports all cash-coded journal entries for a date range.
!DISPLAY Please give Start and End Posting dates
!VALIDATE Start Date(6),D,C
!DEFINE St-Day(2)St-Mth(2)St-Yr(2) = Start Date(6)
!VALIDATE End Date(6),D,C
!DEFINE En-Day(2)En-Mth(2)En-Yr(2) = End Date(6)
:COMMENT -- Extract cash-coded entries for date range --
:RUN TRANSF.PUB.SYS
FROM DBACCS.PUB/PASS,5,JN-JOURNAL TO $NEWPASS &
B25 GE "%St-Yr(2)%St-Mth(2)%St-Day(2)" AND
B25 LE "%En-Yr(2)%En-Mth(2)%En-Day(2)" AND B96 NE " "
EXIT
:COMMENT -- Sort extracted file prior to printing --
:RUN SORT.PUB.SYS
INPUT $OLDPASS
OUTPUT $NEWPASS
KEY 16,10; 9,3; 26,6,DESC; 97,1; 107,6,DESC; 1,6
END
:FILE FSJRNL=$OLDPASS
:RUN PCASHBK
:EOJ
```

Note:
1. Dates are broken down into day, month and year via DEFINE commands so they can be reversed for field comparison.
2. Further selections can be added by simple changes to the input parameters to the extract utility.

INTERNAL PROCEDURES - SYSDUMP EXAMPLE

This next example is not a user application but has been included to illustrate the wider use of a batch interface for internal procedures. It should have some relevance to all H.P. users since it concerns the use of SYSDUMP. We back up our entire system weekly (zero dump) and do partial (date) dumps daily. In order that we can recover quickly, the partial dumps use the date preceding the last full back-up. This also gives us two chains of recovery should we encounter a tape read failure on, say, the last zero dump tape.

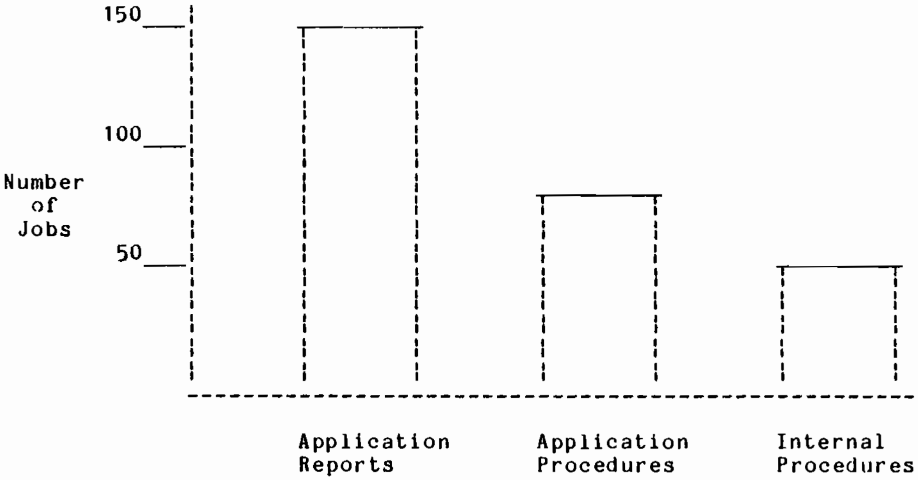
We use a FIJI job to ensure the correct entry of dates. This avoids the possibility of having a 'broken' recovery path due to operator error. Futhermore, it is simple to enhance the job. In fact, the full live job logs the date the dump was taken and the dump-date used via Editor to provide a complete record of all dumps taken.

```
:JOB SYSDUMP,DUMPER/SECURE.SYS
!OPTION D
!DISPLAY ******     DAILY SECURITY DUMP    ******
!DISPLAY    FULL (FRIDAY) DUMP REQUIRED?
%--"Y" OR "N" (1)[Y][N][ELSE=!ERROR Enter "Y" or "N"]
:TELLOP LOAD FIRST SCRATCH TAPE TO RECEIVE DUMP
:FILE LP;DEV=LP,13
:CONTINUE
:LOG LOGIDLD,STOP
:FILE TAPE;DEV=7
:SYSDUMP *TAPE,*LP
N
%--"Y" OR "N" (1)[Y=0][N=!USE DUMPDATE]
@.@.@
Y
:EDITOR
T HOLDDATE
%% Update Hold-date if Date Dump
%--"Y" OR "N" (1)[Y=!GOTO ZERO DUMP]
R 1
%MONTH(2)/%DAY(2)/%YEAR(2)
K
!GOTO CONTINUE
%%ZERO DUMP
%% If zero dump re-set Dump-date to last Hold-date
K DUMPDATE,UNN
%%CONTINUE
EXIT
:TELLOP *** SYSTEM DUMP COMPLETED ***
:LOG LOGIDLD,RESTART
:EOJ
```

Note:
1. The USE command includes the contents of the file DUMPDATE in the output job.
2. The OPTION D command places the machine date in internal parameters 'DAY(2)','MONTH(2)' and 'YEAR(2)'.

USAGE PROFILE

The chart below gives a breakdown of our current usage of the interface within a cross-section of our live systems:



Jobs within the Application Reports category mainly employ RPG or ASK.

Application Procedures include Daily, Monthly and Yearly closings, archiving of data for production of microfiche and loading/unloading operations to interface with external systems on remote sites.

Internal Procedures include a high proportion of infrequently run jobs such as recovery procedures. We have encouraged the use of parameterised jobs in preference to executing tasks online since these can be preserved for future use and thus help avoid "re-inventions of the wheel".

COMPARISON WITH OTHER METHODS TRIED

Quantifying the benefits of using utilities with an interface against other technical approaches is almost impossible. Nevertheless, having witnessed computer manufacturer's statistics regarding machine capability over the years, I feel less inhibited about presenting my own subjective findings.

The problem of directly comparing methods is that unless the same task is done twice no satisfactory benchmark can be made. However, recently we did have occasion to replace a report request program module written in COBOL with a FIJI command file as a result of a major re-design task on one of our databases. The original program controlling user selections was approximately 1000 lines of code having taken about a week to develop. In this particular case, starting afresh with the interface job was quicker than amending the existing code. The replacement command file contained about 100 lines and took three hours to write and test.

Another approach we have used in the past provides an interesting contrast in design philosophy. This was the use of a common 'selection criteria' for reporting. It attempted to provide users with a single comprehensive set of selections (using several forms-fill screens) that could be applied to any report within the application system. But, as is so often the case, user wants do not always reflect needs. In practice, relatively few selections are found to be appropriate to any particular report. Coupled with this, new needs often require selections not included in the original design.

In preference to changing already complex code, new report selections have been added using FIJI. This underlines its ablitiy to keep pace with changing needs as they arise - an important user requirement not easily met using a fixed selection method.

SUMMARY

The advantages we have gained from using a user interface can be summarised as follows:

1. Provides a sufficient degree of user-friendliness to enable utilisation of system tools that might otherwise prove unacceptable to the commercial applications user.

2. Keeps utilities firmly in the hands of the trained analyst who is best able to apply them effectively in solving user requirements.

3. Encourages a modular approach to technical solutions and thereby:
    (a) reduces their complexity
    (b) enables changes to be made quickly and easily
    (c) promotes portability

4. Maintains a consistent "front end" and assists in upholding software standards.

Taken together, these benefits mean fast and cost-effective solutions without sacrificing the quality of applications software.

CHOOSING YOUR OWN INTERFACE

Unlike rapid development aids for online software, utilities suitable for offline processing are comparatively cheap. Likewise, a number of job interfaces offering parameterised interactive entry are available for purchase at moderate expense. One that should be familiar to you is JES offered free in the Contributed Library.

The decision of which to choose will be based on assessing the quality of the user interface required. JES, in common with the majority of job initiators, is mainly a job scheduler and has a limited user interface capability. We have chosen to use separate programs to perform the tasks of interfacing and scheduling. This has enabled us to fully develop these functions independently. Hopefully, in presenting the particular way in which we have employed our interface we can offer constructive guidance to other commercial users.

SOLVE YOUR PROBLEMS WITH IMAGINATION

by
Erik Wendelboe
Hewlett-Packard A/S
Denmark

Abstract.

When running a large HP-3000 installation, you sometimes run into odd problems, for which there are no pre-made solutions, or at least none that you are aware of. When you run into one of these problems, you can just solve it, OR you can try to use your imagination, and solve the problem in a more general way. If you try this, you might end up with a new utility, which can be changed and enhanced, with your changing needs. The paper will discuss how and when to make a utility from a specific problem, and present some of the utilities we have made.

Which problems ?

When developing software systems you focus on your customers needs, I hope, setting standards and making strategies on how to solve all his problems in the best and most reliable way. Thus focussing on all the forseeable problems, you allocates reasonable time to solve each problem, but you will always have severe difficulties in allocating time for the non-forseeable problems, because how much time do you allocate to problems you don't know about ?

Even if you, from your earlier experiencies, try to allocate e.g. 30 % extra time for these problems, you will have a hard time getting extra time approved, in advance, to solve problems that only MIGHT occur. Even if you did get the 30 %, it would not solve the problem, because the worst problems always occurs late at night, when your first big demonstration is due to take place the following morning at eight. In this situation you just grabs bits and pieces from existing programs, to build a program, to solve the problem in the fastest possible way. You know, at the time you are making the program, that it is completely non-standard, but it doesn't matter, because, 'It shall only run this once'. When the problem continues to exist, you continues to run this program, because it does solve the problem. By doing this, you have sneaked a monster in your own backdoor, and as times goes, it gets worse, because nobody knows exactly what the program does, as it is undocumented, and nobody wants to touch it, or even change the environment in which it is executed, fearing that any slight change, will alter the execution dramatically.

The problems, I am referring to, is all those small problems that occurs, when developing computer systems. It can be erroneously updated data, you want to correct, without entering the lot again, it can be problems with performance on the computer, forcing you to make changes in the structure of your system, without too many changes in the programs, or it can be simple misunderstandings, at any level, detected too late in the development process.

These mentioned problems are, as far as I can see, in general, solved too narrowminded. This leaves those programs worthless, if the problem changes just a tiny bit, forcing you to make another program, and another one, and ..., leaving you with an enormous maintenance burdon.

The solution is to make a utility, every time you come across a problem, where you can imagine similar problems, to occur in the future. This can be very difficult to determine, because the time when you can be certain, is the the time when the problem reoccurs. Realizing this, there is only one way to force you to make the right decision, every time. It is to make it EASIER to make a nice clean utility, than to make a big dirty one-time-program. We have made a 'skeleton-program', the MAINLINE, which does make it easier, and furthermore gives you some nice extra features for free.

The 'problem-driven' utilities are, fortunately, only part of the utility development. Another, and more creative, part, is the 'it would be nice to be able to.....' part. To develop utilities this way takes at least two people, and two people who knows each other well, because it takes a long time, via discussions, to decide what it is, that you are about to make. However, the time invested in discussions like this, is always invested wisely, even if you do not make the utility after all, because they forces you to look upon your daily work, with new eyes.

When developing utilities this way, you must always remember to emphasize the benefits of your new utility, as most people do not value the beauty of a program, just the output from it.

The MAINLINE.

I don't know how old the original mainline is, or who wrote it, but the version I have is based on a program by D.David Brown of EASY software, where on both David and I have added a piece here and there.

It is a SPL-program, but the tecnique itself is not bound to SPL in any way, except that you can make things on the 3000 in SPL, which you can not make in other languages. Apart from this, the concept of the MAINLINE is perfectly valid in any other language. The program consists of these modules :

The MAINLINE
Is the outer loop of the program. It will call the initializer, and then continue to call the command procedures, until 'E','END' or 'EXIT' is hit. As a special feature, the program will activate it's father process, (if not the C.I.process) instead of terminating, thus enabling the father process to reactivate the program again, without having to load it.

The Initializer
Will print a banner, initialize variables, and validate the user, and the execution mode.

The Command-reader
Will issue a prompt, read a command, including parameters, and it will echo the entered command, if it is executed in batch mode.

The Command-parser
Will determine a code from the entered command, strip off the command, call the intrinsic MYCOMMAND and pass the information to the command-executor.

The Command-executor
Detects, from the command code from the parser, which user-routine to call. The procedure will call a standard error procedure, if a non existing or non implemented command is requested.

The User-routines
This is the active procedures of the program, the part of the program, which would be the whole program, had you written it as a stand alone program.

The 'Features'
In order to make life easier, we have added a couple of standard features to the program. The 'colon MPE-command' where you can enter MPE-commands, the 'colon RUN', where you can run other programs without leaving the program, (and it will keep up to 8 programs 'warm' and reactivate them on the next run, to save the load) the error-handling routine, which will display error-messages, both FSERR's and your own error-messages, the Help routine, where you only have to type in what you want to put on the screen, and several other, less significant ones.

All you have to do to use the MAINLINE is :
- Copy the source
- Change the banner and the initialization
- Change the Executer to call your routine
- Make the routine
- Change the Help routine
- Add an error-message or two

And you have got a brand new utility.

Examples

GETSTORE
Will take any IMAGE dataset, on a regular storetape, and make a copy, consisting of all active entries, onto a normal MPE-file. The program will take care of deblocking etc. so you end up with the same file, had you made a copy from DBGET mode 2.

GETSTORE was made for DOMI, when they wanted some informations from a dataset within a large database, on a storetape 6 months old, without having to store the current database etc. As it turned out, they never needed the information, but the program has been used several times, in many different situations.

ULTRCOPY

Will copy a file, or a subset of files, (referenced like LISTF, with wildcards @£ etc.) and it will do it at maximum speed. (It will copy 50000 sectors in 1 min. using 4 CPU-Sec.) I made this program for EASY software, to utilize some of the fast I/O-routines David Brown had developed for other purposes.

PORTSCAN

Is rather a small utility-system than a program, with several different tasks:

- It control the logon's
- It monitors the CPU-usage
- It streams and monitors jobs

The system consists of two programs. PORTSCAN is the executing program, and it is running, in batchmode, whenever the system is, and PORTUTIL which is the controlling program, run in both batch and interactive mode. The PORTSCAN-utility was made for DOMI, to control the logon's from some remote LDV's, without the hazzle of maintaining passwords, The other functions of PORTSCAN was added, at a later time, when similar problems occured. By controlling the logon's and the CPU-usage this way, rather than the ordinary way, we have a smoother operation, and yet a much better control. The standard user, obeying the rules, has easier access to the system, the 'bad' user, who sees an 'interesting' logon, can not try the logon on his own terminal, the 'terrible' user, who always accesses databases in sequential mode from his terminal, is stopped after 5 min., and we have a complete record of all logon's to the system, we have a record of the CPU-usage on the computer, and we need not to fear that the 'smart' user has grabbed a password here and there.

The logon check.

PORTSCAN scans the JMAT (Job MAster Table) every 5 seconds (can be changed via PORTUTIL) and logs every logon on a discfile, opened shared, which makes it possible to list the file containing all logon's and all PORTSCAN-violations since system startup, at all times. The program checks the logon against a table, containing all legal logon's. It is possible to restrict the logon's with LDV, User, Group, Account, Job-id or any combination. If someone violates the rules it is logged on the mentioned discfile and on the console, it is possible to get a printout on LP, a message to the violator, and to get the violater aborted. (Configurable through PORTUTIL)

The CPU-usage check.
PORTSCAN scans the PCB-table (Process Control Block) every 5 min. (can be changed with PORTUTIL)and adds the CPU-usage for every job and session. All jobs or sessions using more then 30 CPU-sec. (changeable) is treated the same way as with the illegal logon's.

The job-streaming.
Whereas the two above mentioned functions are 'stand-alone' features, the streaming function is integrated to the rest of the systems on the computer. The streaming is activated from PORTUTIL. What PORTSCAN does is to stream all job-files referenced in a streaming file. (The filename is input to PORTUTIL) The streaming can be divided into 8 seperate job-queues, where one job from each queue is run simultaniously. PORTSCAN is monitoring the jobs via the JMAT, so the next job, in the same queue, will be streamed, as soon as the previous job terminates, regardless how the job terminated. The names on the jobfile, are placed in the streaming-file, (merged with standard periodical jobs) by a function in the normal user data entry program, thus integrating PORTSCAN to the user systems.

References :

"Systems Development, projects Life - The practical Experience"
Paper by Carl Christian Lassen at the IUG Edinburgh Conference 1983

"The Way To The Right Solution"
Article by Carl Christian Lassen, Supergroup Newsletter July 1982

"Centralized Database Access"
Paper by Erik Wendelboe at the IUG Copenhagen Conference 1982

"How to get a High Performance Order-file"
Paper by Hanne Hansen at the IUG Edinburgh Conference 1983

"Experiences With The SL Tecniques, Easing The Development And Maintenance Burden"
Paper by Hanne Hansen at the IUG Copenhagen Conference 1983

How to Avoid Saying ":Hello"

Running Online Applications on the HP3000 without access to MPE.

John Parkinson

SIFO Systems Ltd.

HOW TO AVOID SAYING ":HELLO"

A great deal of debate has taken place over the last few years around the need to provide better security and control of use of online computer systems. Most authors have concentrated on the problems of misuse, theft or tampering with data and on the possibilities of misuse of system functions. The literature on this is now extensive, both in general terms, and for specific application areas. A great deal of ingenuity has gone in to the readily identifiable problem areas (such as the storage and transmission of data) and solutions ranging from the elegant to the complex and expensive have been proposed. In some areas, such as banking and finance, where the investment in computerisation is huge, many of these solutions are already being applied singlely or in combination. In other areas the potential problems are only just being identified and many systems do not yet contain adequate or indeed any security features. It would often appear that it is only applications that involve very large systems or very large volumes of data that can justify the overheads of encryption, special hardware and the associated techniques that are generally proposed to enhance security and maintain integrity in use.

Many users are now familiar with applications that run on much smaller systems, and with the increasing trend to smaller and more cost effective hardware, applications will continue to be developed for a much wider range of users. Combined with the extension of telecomunications services, which offers the potential to increase the complexity of data access paths and to extend the range of facilities available, all aspects of the security and control of access problem are becoming increasingly important. There is a continuing neeed for cheaper and more effective security provisions and aplications systems design has more and more to take these factors into account.

Nevertheless, one class of user and one set of capabilities remains generally outside the security debate, although they combine to present as great if not a greater threat to security than the storage and transmission risks that are usually addressed in the literature. These are:

> The user of online processing applications and the computer's operating system.

Clearly the most powerful tool available for potential misuse of a computer system is the software that runs the system. Online applications inevitably make use of the operating system and its utility functions for a myriad of processing tasks, although once within an application the user will not normally be able to get at operating system functions. The area of risk is thus the duration of the "Logon" process (although Control-Y and "Break" are also problem 'features'). This is usually controlled by some form of password/identity restriction which is adequate for a primary level of security, but is subject to many problems in use. The more friendly the operating system the worse the problem, since it is often possible to persuade the computer to tell a user how to beat its own security, in the guise of being helpful!

The use of combinations of passwords (at user, account, group, file or database level) is often resisted by users, who do not generally want to remember more than one code (and often cannot do so) and who will ignore such security or circumvent it by writing down the passwords, usually on a card taped to their terminal!

Indeed, I have seen an installation which had an excellently planned password system and even went to the extent of changing critical passwords at regular intervals. Unfortunately, the System Manager who could not remember passwords would write the new master codes on a large wall board in the computer room so that his staff would not forget them! This is hardly ideal security.

Once a user is logged on, access to operating system facilities is much more generally available. Restriction of capability can help to limit access to some features and tools, but some users will almost certainly have to have potentially hazardous capabilities (e.g. access to privileged mode DEBUG or DISKED 2) and many applications need to be able to use system features to which users should not have unrestricted access. Maintaining a complex pattern of user capabilities, file access types, passwords or lockwords is generally difficult, and even when set up initially, often falls into disuse as users become less willing to maintain the effort demanded by such a system. Even the use of a more sophisticated or personalised security feature (MPEX from VESOFT is a good example) will not necessarily solve this problem.

Even well thoughtout and rigidly adhered to password systems can break down under the attack of a determined intruder and the problem is that, once the system has been penetrated, much or all of the operating system capability becomes available for misuse. Short of totally emasculating the operating system on machines running live applications, which is often infeasible for operational reasons, it is difficult to prevent determined or even casual misuse of the system. Indeed the greatest threat often comes from the "browser" who regards it as a challenge to break through the computers security system, or who just wants "to see what the machine will do"! Curiosity is often the biggest source of misuse.

Sifo Systems has been providing and developing sophisticated online health care management systems for a number of years and originally chose to offer HP3000 hardware because of the high quality of the application development tools, the reliability of the equipment and the facilities of the operating sustem. It soon became apparant however the the very facilities that were of benefit in the development shop could well be a liability in the live user environment. Health care users are especiaslly sensitive to the threat of unauthorised access to data and system facilities and it was necessary for us to demonstrate that we could guard against both inadvertant and deliberate attempts to misuse our applications. In the end, despite providing all the usual password security facilities on users, accounts, and files and audit capability on all usage we decided that the only totally secure approach would be to completely insulate the operating system from user access, so that at the most, a potential misuse of the system would be limited to the functional capabilities of the application modules being provided. These we could control,via passwords, much more tightly than could be achieved for the operating system and at much lower cost. We therefore determined to develop a software system that allowed users to run online applications without being in contact with MPE at any stage from switching on the terminal to finally switching it off (and allowing for the user who never switches off as well!).

Design Concept

The requirements of the new approach were easy to set out. They were:

- The software must detect the presence of a newly powered on terminal without the user pressing a key on the keyboard i.e. no ":" could appear. This is primary "insulation" that prevents the user gaining access to MPE.

- It should be possible to control and monitor activity on all terminals on the system from one point in an interactive fashion.

- A log of all terminal activity should be kept.

- It must be possible to support spooled terminal printers on the system and to control which terminals could use which printers.

- The standard password facilities of MPE should be retained and where appropriate extended.

In the interests of flexibility it was also decided to add the following design criteria to the above:

- Configuration of applications under the system should be entirely file driven using standard MPE facilities.This allowed "soft" configuration of the system by user management.

- Individual users access to system facilities should likewise be file driven and not hard coded.

- The system should allow maximal use of the HP3000 resources so as to allow large online systems to run under it. (We were aiming to run up to 100+ point to point terminals on a single processor system.)

- It should be possible to disable the "Break" function and the Control-Y (Subsystem break) capability of all application terminals so that users could not jump out of an application, even if they wished to. This feature provides the secondary level of insulation and prevents active terminals reverting to session mode by breaking out of the currently active program.

It was clear from the outset that the software would need to be written in SPL, and that we would need to develop a greater degree of MPE expertise than had hitherto been needed. This process took longer than anticipated, due to the relative lack of accessible training and documentation available from HP in the UK. Once the basic knowledge had been gained, however, development of the prototype software was rapid and the initial system was available inside 3 months. This consisted of the following.

- A master control program (CCS) written in SPL and providing full system set up, monitoring anmd management functions.

- An Access control system (ACCESS) written in COBOL which is created as a son process by the CCS for each terminal on the system. This program acts as a framework for all other applications software.

- A terminal spooling system (NSPOOL) that allows VDU's to direct printed output to one or more printers, without recourse to the MPE spooling facilities.

Only one copy of the CCS needs to be loaded to run any number of terminals and applications. The basic information identifying terminals, printers, spoolers and their relationships is held in a text file, built via EDITOR or TDP and may be encoded if desired. As each terminal referenced in this file is switched on it is captured by CCS (in less time than the screen takes to warm up) and the BREAK key is deliberately disabled. The terminal is now isolated from direct contact with MPE until the CCS is terminated or until it is released from CCS using the interactive configuration management functions.

Each terminal that is connected to the system has an ACCESS process created for it as soon as it is switched on. CCS passes all the necessary device and file information to ACCESS via an extra data segment (XDS) . This method was chosen in preference to HP's own interprocess communication intrinsics for efficiency reasons and has worked very well. This XDS is also used to pass status and error messages back to the CCS process, for display on the Master Control Terminal. Control of XDS access is implemented via resource identification number (RIN) locks. (This requires MR capability and some clear thinking to avoid RIN lock conflicts which can hang the system!).

ACCESS handles the password security routines for a user and relates each userid/terminal combination to a matrix of access paths to available applications and intra-applications functions, held in an IMAGE database. This database is maintained via online transactions within the ACCESS module, which are password protected or by QUERY.

ACCESS also maintains a log of all terminal activity and produces an analysis of system usage on a daily and weekly basis. All attempts to use the system, including failures to complete the correct startup sequence, are recorded.

ACCESS also constructs an application selection menu according to the users's password and associated access permissions, so that only applications that the user may access are displayed.

Each terminal may also be associated with one or more spooler (NSPOOL) processes that allows applications output to be directed to one or more online terminal printers. The system does not use the MPE spooling system, but again makes use of an XDS to transfer print data to and from processes. In this way a large number of spooled printers (currently up to 32) can be supported. User applications may share spoolers, according to the volume of online printing being carried out.

Control logic is provided by CCS and NSPOOL to allow for:

- Printer unavailability where output has to be held or diverted to another printer

- XDS overflow in very high output traffic situations

- Load balancing between printers.

Using this system, applications can either:

- Send all their output to one printer or to a group of printers

- Send particular outputs to a specified printer or group of printers (to cater for special stationery needs).

- Share a printer with other applications

Benefits of the System

As well as the immediate benefit of achieving our aim of preventing user access to MPE and its various subsystems, a number of other operational benefits have become apparant as actual user experience of the system has increased. These are as follows:

- The number of processes and therefore process identification numbers needed to run a large online system (60+) users can be greatly reduced. Typically a system is configured as follows :-

- MPE processes	c. 20 PIN's	
- CCS	1 PIN	
- LSPOOL processes	5 PIN's	
- ACCESS/users	1 PIN	per terminal.

  This is achievable since applications are called as subprograms by ACCESS and do not require a separate PIN.

  It is therefore possible to run up to c.100 online terminals under this system (rather than the 60 or so allowed by MPE in session mode) without using MULTIPOINT.

- It is possible to support many more online spooled printers than HP supports using its own terminal spooler.

- It is possible to monitor and manage a terminal configuration online, interactively, independent of the MPE session system. Features provided to do this include a set of comands such as:

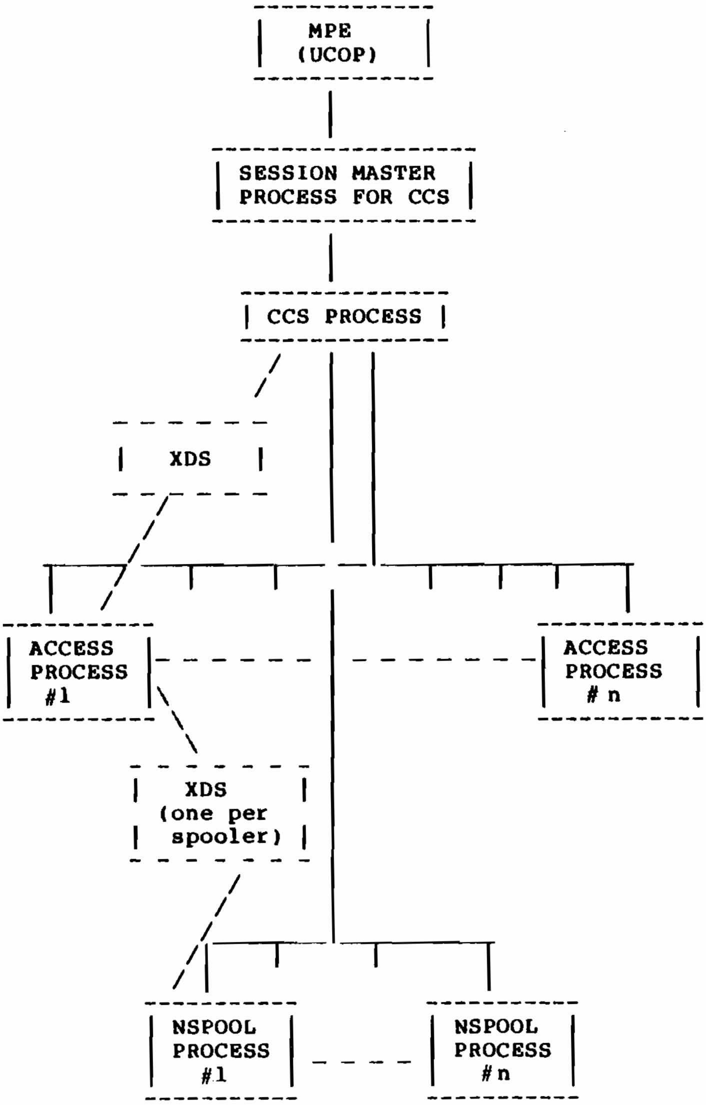
  - UP to add a terminal, printer, spooler or communications line to the system.
  - DOWN to remove an entity from the system.
  - ASSOCIATE to establish a relationship between entities on the system.
  - LIST to show associations.
  - SHOW to list current activity.
  - TERMINATE to stop CCS.
  - HELP to review available commands.

The commands are input via a dedicated terminal (Master Control Terminal, MCT) which cannot be the console. We normally use a 2635 or similar for this, so as to maintain a hard copy log of CCS usage. Access to the CCS functions uses a Control-Y interrupt from this MCT which automatically times out after 20 seconds if no action is taken.

There are, of course, performance factors that have to be taken into account when using the CCS/ACCESS concept. The system works best when a large number of online users (60-100) are using a small number of applications (less than c. 20 max, typically 8-12) when CST usage and code segment swapping can be minimised. Even then, the XDS usage data segment requirements and overall virtual memory means that system tuning is essential for adequate performance on small memory configurations. (We use LOOK/3000 extensively to monitor application activity and IOSTAT to balance disc usage). Our best performance to date is c. 30 users on a 1.5mb 3000/44 processing c. 12000 transactions per hour and supporting 12 spooled terminal printers. Larger systems have been configured but require more memory or multiple processors. We are presently evaluating final configuration details for a system based on a dual 4MB 3000/64 supporting c. 300 VDUS and 100 printers. At this size of configuration, we are constrained by maximum allowable system table values (CST, PIN, DRT etc) from adding more terminals and by disc access bandwidth from processing more transactions. CCS/ACCESS will however remain stable up to configurations of this size provided adequate care is taken when setting system parameters.

It is also virtually impossible to run batch work or to do development on a nearly loaded system when CCS/ACCESS is in use, but this is just as true when using conventional MPE sessions!

DIAGRAM


MPE
(UCOP)
SESSION MASTER PROCESS FOR CCS
CCS PROCESS
XDS
ACCESS PROCESS #1
ACCESS PROCESS # n
XDS (one per spooler)
NSPOOL PROCESS #1
NSPOOL PROCESS #n

Applications programs are implemented as callable subprograms from the Access process.

The system as described above was tested thoroughly at a user site over a 6 month period and has now been installed and running at a number of sites for around two years. During this period, a number of further features have been added. These allow:

- the control of DS links to other HP3000 systems, allowing CCS to run transparently over a multiprocesssor network. We have currently implemented this feature in a 3 processor system;
- the control and use of communications links to other non-HP computer systems. We have implemented links to Honeywell, DEC, CDC and IBM systems;
- the control and use of non-standard peripheral devices, in this case an online plastic card embosser is supported on the system. Other devices could similarly be attached.

These developments have continued to prove the basic robustness and flexibility of the approach, since applications modules do not need to be concerned with the physical or logical configuration of terminals, printers or other peripherals.

For applications designers who do wish to use the capabilities of the CCS/ACCESS system directly (e.g. XDS message passing), a set of special intrinsics is provided as an RL or SL.

Current Extensions to Facilities

With the increasing availability of a range of HP and third party supplied software packages, an obvious extension of the CCS/3000 facilities is to allow users to access such software from within the CCS environment. Our work in this direction was prompted by the requirements of one customer site to run TDP, HPWORD and HPMAIL as well as our applications software on their system. There are some technical problems involved in achieving this mode of operation, since the HP software all assumes session mode working (and hence assumes the existance of file system entities like $STDIN, $STDLIST and so on) and utilise special control sequences (such as CONTROL-Y) to implement some features. Information on technical details is often hard to get or expensive (or both) although products designed at Pinewood are less of a problem, once the right contacts are established! To date we have managed to implemented the majority of the required features to allow access to TDP and HPWORD, although the system is less robust in use than we would like, and some problems do remain. We expect to have these sorted out by the end of 1983, and to be able to fully support microsystem based workstations (HP 120/125, series 20 and DIRECT 1025) by this date also. The release of the new Series machines (3000/42 and 3000/48 in particular) plus the capabilities of MPE-V have removed many of the previous resource constraints on the system and we are now able to offer our customers even larger systems capacity and performance via CCS/3000.

RELATE/3000 - RELATIONAL DBMS FOR THE HP3000

INTRODUCTION

All computer users store data. Any file represents a database to its owner. Experienced users of the HP3000 will be familiar with the different types of file storage available - MPE, KSAM, IMAGE. What follows is an introduction to another type of file structure, and more particularly the Database Management System available to HP3000 users called RELATE/3000.

The Relational Database system will revolutionise data organisation and accessibility.

As a starting point let us look at the traditional storage methods, or "data models".

Table File - also called "Flat" file consisting of records made up of one or more fields of data. The link between fields is fixed by their location in the record.

More advanced data storage methods attempt to minimise the duplication of data and to make more complex linkages between data fields possible. The possible linkages may be -

one : one

e.g. Item code linked to description in an Inventory file, the Table file

many : one

e.g. Personnel in a Department., actually the commonest commercial file type

many : many

e.g. University Students linking to the Courses they all take.

HIERARCHICAL DATA MODEL

Introduced in the early 1960's, this represents data as a hierarchy. It is easy to understand data structures like this. Examples are company organisation charts, or Bill of Material explosions. It introduces the concept of storing not only data, but also the need to maintain the linkages between data in some form of file. The disadvantage of most Data Base Management Systems built on this type of data model is that once built, the file structure is difficult to rearrange or modify.

NETWORK DATA MODEL

This dates from the early 1970's and was a modification of the Hierarchical model. It was more storage efficient than earlier models - and more difficult to understand. Often, although data that is needed is present in a data base, there is no linkage for it and the user cannot therefore obtain it without restructuring and redesigning the network. Even in the best DBMS based on this model, some knowledge is required for the user to be able to navigate his way around the data.

RELATIONAL DATA BASE

Relational data models represent data in two-dimensional tables that are easily understood by non-technical users. Each table file is self-contained or can be dynamically linked with other files as required.

Quote from Dr. E.F. Codd, IBM Research Fellow:

> "A true relational system requires all information to be represented at the logical level as values in TABLES. There must be NO USER-VISIBLE NAVIGATION LINKS between tables and the system must support ... at least the SELECT, PROJECT and JOIN ... operators of ... relational algebra"

What is meant by these terms can be illustrated by looking at a small relational database which tracks shipments of parts. The data is stowed away on the machine in two "flat" files.

SUPPLY - containing the Shipment no., Supplier Name, Shipment Status Code, and the Supplier's City.

SHIPMNT - made up of the Shipment no., Part nos. and Quantities of parts.

Here is the SUPPLY file made up of rows and columns.

```
>OPEN FILE SUPPLY
>PRINT

£LINE SN SNAME      STA CITY

    1 S1 SMITH      20  LONDON
    2 S2 JONES      10  PARIS
    3 S3 BLAKE      30  PARIS
```

to "SELECT" from this file means to extract a row, or record

```
>SELECT @ WHERE SN = "S3"
>PRINT

SN SNAME      STA CITY

S3 BLAKE      30  PARIS
```

To "PROJECT" means to extract a column, or field

```
>SELECT SN
>PRINT

SN

S1
S2
S3
```

Another term which Dr. Codd mentions in his definition is JOIN. This begins to illustrate the power and the possibilities of a Relational database. Physical files can be linked together to create virtual files. To the user they can appear to exist and can be accessed, interrogated, - even updated, but in fact the data resides in different places and is not linked until the JOIN operation is set up.

```
>OPEN FILE SHIPMNT
>PRINT

£LINE SN PN  QTY

    1 S1 P1  300
    2 S1 P2  200
    3 S1 P3  400
    4 S2 P1  300
    5 S3 P2  200

>SELECT SHIPMNT.@, SUPPLY.@ WHERE SHIPMNT.SN=SUPPLY.SN
>PRINT
```

SN	PN	QTY	SNAME	STA	CITY
S1	P1	300	SMITH	20	LONDON
S1	P2	200	SMITH	20	LONDON
S1	P3	400	SMITH	20	LONDON
S2	P1	300	JONES	10	PARIS
S3	P2	200	BLAKE	30	PARIS

Early Relational Database systems were developed by research mathematicians. They got the reputation for being inefficient and slow, but this is no longer deserved. The RELATE/3000 DBMS which we shall look at now is by no means the product of theoreticians. A year ago the "Small Systems World" magazine examined RELATE and compared it with Relational DBM systems on machines other than the HP3000, notably DEC PDP-11 and IBM Series/1. Unfortunately there was no comparison made of performance, but the system compared very favourably with others in the marketplace.

Because of the use of B-tree indexes in RELATE, file access times are the same whether files are a thousand or a million records long. The attraction for new users is the friendly interactive Command Language, and if access proves slow then it is because the system is being asked to make linkages which would be difficult or even impossible in other DBMS. What makes the RELATE system particularly interesting is that it can use ordinary MPE files, KSAM files and IMAGE database files not only as input, but can also update, add to and delete from these files as well as its own RELATE/3000 files.

COMPARISON OF RELATIONAL DBM SYSTEMS

RELATE/3000	PRODUCT	ADABAS-M	PROMPT	INGRES	ORACLE
$18.500	PRICE	$40.000	$6.000	$30.000	$24/96K
$2.775	ANNUAL MAINT.	$4.000	$600	$3.000	1/10th
HP3000	MACHINE	PDP-11.VAX	Series/1	PDP-11.VAX	IBM
YES	INPUT NORMAL DATA	YES	YES	YES	YES
interactive	DATA DEFINITION	hybrid	-	i'active	i'active
10	USER FRIENDLY	6	-	10	10
8	RESTRUCTURE/REFORMAT	6	4	8	8
2	DATA DICTIONARY	3	-	10	10
indexes-yes data-no	SELF-REORGANISING	yes	no	no	yes
no	CRASH-PROOF	yes	no	yes	yes

RELATE/3000

gives the user an interactive language of 46 COMMANDS

general format:

```
        (RANGE)  VERB  (PARAMETERS)  (CONDITION)
                 ----
```

e. g.

```
10/100 CONSOLIDATE QTY BY PN TO ORDERS
       -----------

       PRINT  PN, PNAME, COLOUR FOR WT<16
       -----
```

RELATE/3000 COMMANDS

DATA DEFINITION - CREATE
PURGE

DATA ACCESS - OPEN
CLOSE
SELECT
SET

DATA ENTRY - ADD

USER ASSISTANCE - COMPILE
DELETE
ERASE
EXECUTE
HELP
IGNORE
NOTE
PAUSE
RECOVER
REDO
REORGANISE
SHOW
SORT
SYSTEM
TERMINAL
:mpe

DATA MANIPULATION - COPY
CONSOLIDATE
CHANGE
FIX
LET
MODIFY
UPDATE

REPORTING - COMPARE
LABEL
PRINT
QUIZ
SUM

SECURITY - ALLOW
DENY
DISABLE
DISALLOW
ENABLE
LOCK
PERMIT
UNLOCK

TRANSACTION CONTROL - BEGIN
ABORT
COMMIT

RELATE/3000

gives the user

HOST LANGUAGE INTERFACING * to

- COBOL
- FORTRAN
- PASCAL
- SPL
- BASIC

Intrinsics:

OPEN and CLOSE FILES	RDBINIT RBCLOSE
RECORD-AT-A-TIME OPERATIONS	RDBADD RDBUPDATE RDBDELETE RDBREAD
READ-BY-INDEX	RDBPOINT
RETURN CURRENT STATUS	RDBINFO RDBERROR
MATCH SYMBOLIC/VARIABLE FIELDS	RDBBIND
PASS RELATE COMMANDS	RELATE

DEMONSTRATION

To finish with here are a couple of examples of RELATE/3000. The first is simple inventory control application which will demonstrate the way in which data is entered and manipulated. The second is from a legal billing system and will demonstrate the flexibility of reporting which is possible.

SUMMARY

Relational database technology is available now. To summarise, RELATE/3000 gives:

1. User-access to data with easy-to-learn command language. Training costs are significantly reduced.
2. Programmer-productivity through the power of system, and because users can do more for themselves.
3. Improved flexibility over other database techniques now available.
4. Reduced system development times because existing data can be readily converted or accessed without major reconstruction.

DATA BASES ON MICRO-COMPUTERS.A REALITY.

Michel KOHON

DataSoft International

2, York Road

MAIDENHEAD, Berks.

ENGLAND

The topic of this presentation is two-fold :

How did we design a Data Base Management System for Micros,

How will you design Data Bases on Micros.

I. CREATION.

Micro-computers really came to the professional market about 2 years ago with Hewlett-Packard,Xerox and IBM decision to produce their own micros and to use the very popular CP/M Operating System.

The already visible success of Apple Corp was mainly due to a spread sheet called VISICALC.This piece of Software was so user-friendly that hardly any training was necessary.Both HP and IBM decided to buy the product and to offer it with their own micro.

The mere fact that major manufacturers entered the micro-computer market created a very new situation.In a certain way it 'established' micros.Large corporation managers started using them or obliged their staff to do so.

At the same time, most DP managers did not see the real benefit of micros since there were no tools for them such as languages or Files Management Systems.

The only language provided was BASIC and many users started to learn it.I did it as well and must admit that it was fun before reaching the point where I had to modify my programs.Even structured BASIC is not easy to maintain, and I could not envisage my programmers wasting their costly time struggling with such an Amateur language.I decided instead to put into practice PASCAL.

PASCAL has several convenient features for commercial applications.
The first is its portability since that all existing PASCALs are based on the same Set of standards.You can make a program on a

MICRO and transport it on the HP/3000 with a good probability that it will work.

The second is its simplicity which makes it easy to learn from practical examples.Unfortunatly PASCAL is often presented as a theoretical, mathematical language due to its origins.

I compared several compilers running under CP/M and decided to go for a very strict one offering the same features than both HP/1000 and HP/3000 compilers.I made parallel tests both on the HP/3000 and on the micro to assess the level of compatibility.

Tests were very positive in the sense that most programs were running on both machines without any major modification.

However I reckoned that my efforts were useless since I had no Data Bases and was using sequential files.

I started looking for a DBMS accessible from my programs.I received all possible information one can get.I tested every single DBMS I could have access to.The results were very much disapointing.

Two of the products were not DBMS whereas called so.They were offering no direct way to be accessed by programs.The structure was hiden.One could not access several files or Data Bases at the same time.Failures would destroy the integrity of the Data with very poor ways of recovering.Security was banned.

A third one was a true DBMS but so complex that I could not understand all the features and I could not figure out how to make it work on 64 Kbytes.Moreover the price was so high that I was unable to justify it to my management!

I had no alternative but to write a DBMS aimed at micros .It was logical to make it compatible with the beloved IMAGE.

I called it MIRAGE since its was more a dream than anything else.

Since this is not a vendor presentation,I will introduce MIRAGE as a concept rather than as a product.

PREVAILING CONCEPTS OF MIRAGE.

1. PORTABILITY.

MIRAGE is completly written in PASCAL.The choice of the language was based on my paramount objective of transportability.It is not enough to transport programs if you cannot transport your Data Base structure.Since MIRAGE uses normal sequential files which exist on any machine, I can transport MIRAGE on any system with a PASCAL compiler.These systems could be MINI or MICRO and eventually MAINFRAME.

2. LOGICAL ORGANISATION.

Like IMAGE, MIRAGE is organised in a NETWORK.Today's DBMS science offers us three models which are :

. HIERARCHICAL

. NETWORK

. RELATIONAL

The hierarchical is lacking flexibility.The relational is lacking ressources.NETWORK is a good trade-off.

Whereas IMAGE is a two level NETWORK, MIRAGE is multi levels.IMAGE offers MASTERS and DETAILS.A DETAIL cannot point to an other DETAIL.When a DETAIL points to an other DETAIL the DBMS is a Multi-Levels Network.There is no indication in the design of MIRAGE regarding the maximum number of levels it may support other than the 99 possible files.

Later on we will come back on the actual usage of multi-levels structure since IMAGE designers have been restricted to the two levels organization.

3. PHYSICAL ORGANISATION.

One specific feature of MIRAGE is the physical separation of the actual DATA from the POINTERS which are necessary to navigate in the Data Base.I said physical.In a DBMS, one should always separate the Logical side - the way users and programmers will look at the Data Base - from the Physical Side - the way Data and Pointers are actually organised on discs.

IMAGE Data Sets' hold both Data and Pointers.They are stored together in what is called a MEDIA record.On MIRAGE they are separated.Three considerations led to that split.

The first was the current Micro computer lack of Multi-volumes management.One physical file must be on one physical disc drive.By storing together Data and Pointers I would not have been using efficiently the disc storage.

The second was Data integrity .One can always rebuild its Pointers from Data.Actually MIRAGE provides recovering utilities which are essential on Micros.

The third was flexibility.The separation offers the possibility to build a Data Base out of existing Files which is interesting in the following two situations :

.You have existing files which you want to organize into a Data Base schema.

.You already have a Data Base and you want to include a new file in it.

To illustrate the second situation I will refer to what happened in one of our installations.
A system was in operation and our client asked for a new program.This program was to operate from a new Data Set as well as from the old Sets.We created a new Data Base which was consisting of the existing Sets plus the new Set.The separation of Data from pointers made our new Data Base invisible for the old Data Base users.It meant as well that two Data Bases may share the same files.

4. OPERATING SYSTEM INDEPENDANCE

MIRAGE had to be independant from the OS since no one can predict which one,if any, will prevail in the future.Moreover, the existing OS are so poor that you hardly rely on them.MIRAGE independance is insured by the language it is written in and by the use of 'flat' sequential files .MIRAGE assumes that the MURPHY's law will be in action at every moment to destroy files or stop the machine in the middle of a DBPUT.MIRAGE tries to decrease the probability of such nasty actions by different means such as closing or copying files.MIRAGE tries to decrease the seriousness in offering recovering utilities.

5. STRUCTURE FLEXIBILITY

The possibility to change the Data Base structure was originally included in the Design.MIRAGE allows redefinition of ITEMS,FIELDS,PATHS,RECORDS,DRIVES,SETS and PASSWORDS.Most of ADAGER features are included in MIRAGE since we all know that there is no such thing as a STABLE Data Base.

6. STRUCTURE and DATA DEFINITION INDEPENDANCE.

Data Definition is achieved by creating Items and Fields.
An Item is a specific Data Entity which holds its characteristics such as size,type,securities,..etc.
A Field is an Item within a Data Set Record.It holds all the items specifications plus the position where it is stored in the Data Set.

When you make an IMAGE Schema, you define the Items,and then the Data Sets.An IMAGE Data Set definition indicates what will be the Links and what will be the Fields.To illustrate you might say that 'ACCOUNTNO' is the key of a MASTER which is linked to three DETAILs.By doing that you are in fact doing two different actions.

> The first is the selection of 'ACCOUNTNO' out of the list of Items.
>
> The second is the linkage of 'ACCOUNTNO' to 3 DETAILs.

The first action is to FIND and the second to REPORT a Set of FIELDS.The best illustration is to think at the way QUERY works.

Let's imagine that you have a ten characters long KEY which is in reality composed of two parts like 'MAIN ACCOUNT' and 'SUB ACCOUNT'.To save space on your report you want to report the 'SUB ACCOUNT' only.IMAGE obliges to program this edition by yourself.

MIRAGE design allows to do it without programming.For MIRAGE, the two fields are separate entities, whereas the concatenation of them is the KEY.

Other features are possible such as OVERLAPPING, or SUB-KEYS.
A KEY can be defined across several FIELDS, or across parts of FIELDS.A KEY can be defined inside a FIELD.

These features may be very convenient in term of Data Base flexibility.The structure of Fields can be changed without changing the Data Base Linkage Structure (Keys and Paths).

Although some of the presented concepts are different from the concepts used in IMAGE, I would like to stress that one can use MIRAGE in exactly the same way one uses IMAGE.

The second part of this presentation will be addressing System Designers.How can they design Data Bases which can be integrated in the HP/3000, and how they can interface MIRAGE and IMAGE Data Bases.

II. USING MIRAGE

It is rather essential to appreciate that there is no real difference between an IMAGE Data Base design and a MIRAGE one.The fact that the Data Base will be on micro does not change the nature of the solution you want to provide your users with.

However the designer must clearly know wether whereas he wants to stay compatible with IMAGE or not.The decision not to use multi-levels Data Sets both MASTERS and DETAILS is a painfull one as soon as you have already tried it,but this decision has to be made if compatibility is at stake.

Let me explain now what is a MASTER/DETAIL Data Set.

Since Data and Pointers are physically separated you cannot tell as easily as with IMAGE when a Data Set is a Master or when it is a Detail.The Data Base structure permits to view it either as a Master or as a Detail.

To illustrate this ,we will consider the following schema.

A Data Set is pointed to by a KEY in a one-to-one way.For both IMAGE and MIRAGE, this Data Set is a MASTER.

A Data Set has several Keys, each one linked to a Master, and more than one record have the same Key value.For both IMAGE and MIRAGE, this Data Set is a DETAIL.

A Data Set is pointed to by a KEY in a one-to-one way, AND has several keys linked to Masters.For IMAGE it is an impossibility,whereas for MIRAGE it is a MIXED Data Set.

With MIRAGE this Data Set can be seen either as a MASTER or as a DETAIL.It means that you can get access to it either by using a DBFIND/DBGET 5 ,or by using a DBGET 7 depending on the Key you specify.It is relevant to say that your Updating programs are likely to use DBGET 7,saving programming time and disc accesses.

This feature cures as well the problem of Detail Data Set records which must not have duplicates for one of their Keys.IMAGE obliges to declare a Manual Master to control that Key, and to programmatically make sure that each new record has a different Key value.MIRAGE makes that control for you.

Let's take the example of an accounting Data Base with a Master 'ACCOUNTS' Set and a Detail 'VOUCHERS' Set.

Each entry in the so-called 'Detail' has a unique 'VOUCHER No' which is the Master Key.Details Paths like 'DATE', 'JOURNAL' might be present.The fact that you want to stay compatible with IMAGE means that you must avoid that design at all costs.One possibility might be to start with a pure IMAGE structure and then to add the Master view.That is for each and every one to decide for himself the level of compatibility he wants.

After the preliminary question of whether you want to stay compatible or not,the second question is to decide if you want or not to link your micro Data Base to the Mainframe one.

We know that the concept of the integration of two different Data Bases is one of the main topics now under evaluation in most DP departments.MIRAGE compatibility with IMAGE addresses directly the problem.

The way to do it today is to transfer an IMAGE Data Set (or part of it) into MIRAGE and vice versa through temporary files.Both the extractions,and the insertions have to be programmed.The transfert it-self can be done with existing utilities generally delivered by the Micro manufacturer.

Designers can think of applications where the local user wants to make his own input with a control against his CUSTOMER file.Time to time ,transactions will be transmitted to the mainframe for consolidation.Designers may allow the user to make new CUSTOMERS creation, whereas the mainframe will control duplicates.

On the other way, designers can use the mainframe to dispatch Data to a Network of Micros scattered throughout the country.Grain Silos,for example, may receive Delivery Orders to be executed that day without having to be linked constantly with an overloaded mainframe.

However the way to do it to-morrow will be more straightforward since MIRAGE will have most of the DS capabilities.DBGET and DBPUT on IMAGE Data Base will be monitored from MIRAGE.

Another way designers might think of using Data Bases on Micros is to simulate the mainframe.You may design, write and test an application on a Micro, and then transfer it on the HP/3000.

We all are aware that Production and Developments do not go well together on an heavily loaded HP/3000.In fact they do not go together at all.They are competing for scarce ressources.

If your programmers use Micros, they can develop and test without using the HP/3000 ressources whereas they can still get access to it whenever they want since Micros are also normal terminals (not all of them).

Programmers can share an HP/3000 based library of programs and only transfer those they need for tests on Micro.You may allow programmers to take home their Micros in order to complete in due time their programs.It also means that developments can continue during SYSDUMPs,RELOADs or SYSTEM FAILUREs.Moreover, fast full-screen Word Processors can be used to write programs and documentation.
Training of Junior programmers and users can be achieved using Micros.Learning IMAGE with MIRAGE, or QUERY with MENTOR could be done without the DPM's fear that one could make SERIAL READ on

Data Sets, or DELETE them.At least, if one does it ,he will bear full responsibility for it and it will not have any serious consequences.

Program demonstration could be done with micros since they tend to be portable, if not transportable.It is easier to move around a stand-alone Micro, than a dumb terminal, a MODEM, a line, and a working HP/3000.

To terminate this presentation I would like to present an exemple of a Multi-level Data Base using Mixed Data Sets.

Let's look at a very simple IMAGE like ACCOUNTING Data Base.We have a MASTER which stores the Chart of accounts, and a Detail for Daily entries.There is one unique Entry in the Master for each main account, and this Master is linked to the Detail.

Let's imagine that the Detail is also linked to an other Master Data Set throught a Path consisting of the registration date. Every one can notice that this design is a pure IMAGE one.

If we want each Detail entry to have a unique 'NUMBER',IMAGE will obige us to create a Manual MASTER bearing only that unique 'NUMBER'.It must be a Manual in order to have the management of the Set under its control and avoid duplicates.

A multi-level DBMS permits us to make that link automatically by mentionning that the 'NUMBER' is the Master key of the Voucher Data Set.

Let's now assume that each one of these entries, in our Voucher Data Set, is monitoring entries in a sub-voucher Set.It is rather convenient in a Payable-Receivable system for automatic matching.

Invoices are registered in the Voucher Data Set, as well as Payments.However Payments will be also recorded into the sub-voucher Set with the Invoice number.This system permits you to list all partial payments against any invoice without having an interface Data Set.

Many other practical solutions can be considered when talking in terms of multi-level Data Bases and the skilled IMAGE designer will surely take advantage of them.

However the more levels you add, the more complex the design will be.You may even end up locking yourself.

The possibility to define more than two levels does not mean that you to abuse of it and fall into the 'galoping elegance' syndrome.

III. CONCLUSION

Data Base for Micros is a reality of today.Its modern design is taking advantage of past IMAGE experience and bring a link for tomorrow's integrated systems that you will from now-on start thinking about.

Computer
Museum

NUCLEUS: A System Management Tool

W. Gary Sitton
BI-TECH Software, Inc.

ABSTRACT:

Multi-user HP3000 sites are typically faced with needs to manage the following information:

1. User capabilities to access information at the data base, data set and, most critically, data entry levels.

2. User job launching and running capabilities.

3. Allowable user log on passwords, days, times, LDEV's and attendant user specific device class assignments.

4. Codes used commonly throughout applicational systems.

5. DBA access passwords to data bases.

6. Financial subsystem audit trails.

7. Dialogue and JCL required for standard menu-driven user interaction which results in the launching of a batch job or the activation of a process.

8. Error codes, descriptions, severity and corrective advice.

9. Report and screen headings, formats and identifications.

NUCLEUS is a system which supports a structure that addresses the above stated needs. Specifically, the structure supported by NUCLEUS is a 3NF IMAGE/3000 Data Base named ROOTDB.

INTRODUCTION

The purpose of this paper is to describe NUCLEUS, a system designed to address several problems commonly found in a multi-user environment where several independent software systems are operational. NUCLEUS is a session/job management tool; a software tool; and, a system management tool.

As a session/job management tool, NUCLEUS is designed to provide a 'cocoon' for the user from log-on to log-off. When the user logs onto the system, a LOGON UDC invokes a program, NULOGON, that establishes a brief dialogue with the user which results in the creation of a session temporary file, USERID. This file contains information about the user which may be accessed from any application program. Upon exiting NULOGON, the UDC places the user under the control of a FATHER process called NURUNJOB. This process conducts dialogue with the user which results in either the launching of a job or the interactive execution of a JCL sequence. Thus, NURUNJOB conducts user dialogue which either STREAMS a job or ACTIVATES a SON process. The user never leaves NURUNJOB, except to log off the system.

This method of session/job management provides a controlled environment in which all users have only one UDC, the LOGON UDC; all user desired functions are initiated through a dialogue structure. With the ability to designate tasks as interactive or streamed, the system designer is given great flexibility in terms of user control. In addition, the DP Manager has a stable framework from which to control methodologies employed in user interfaces.

As a software tool, NUCLEUS manages error messages, report and screen headings, common codes and all user interface information. The centralization of such management frees the programmer from many tedious tasks; provides a regular structure which makes program maintenance easier; avoids inconsistencies which result from independently written user interfaces; and, provides a convenient and fast method for the development of application specific user dialogue. For the DP Manager, this provides a common structure which can be used by all systems; i.e., it facilitates centralized control.

As a system management tool, NUCLEUS manages DBA access passwords to data bases, keeps track of useful month-to-month information and provides a record of application software abnormal terminations.

NUCLEUS may be thought of as a system which manages the following information:

1. Application system audit details;

2. User descriptions, data base access capabilities and process running capabilities;

3. Structures necessary to conduct user dialogue which results in the initiation of JCL execution;

4. Common coded information;

5. Report and screen headings; and,

6. Error messages and recovery advice.

NUCLEUS performs information management functions through the use of the IMAGE/3000 Data Base, ROOTDB, shown in Figure 1.

```
_________ _________ _________ _________ _________ _________ _________
|  AU-  | |  CD-  | |  US-  | |  HD-  | |  QS-  | |  ER-  | |  NQ-  |
| AUDIT | | CODES | | QUSER | | HEADS | | QUEST | | ERROR | | NEXT  |
 \-MSTR/   \-MSTR/   \-MSTR/   \-MSTR/   \-MSTR/   \-MSTR/   \-MSTR/
  \   /     \   /     \   /     \   /     \   /     \   /     \   /
   \_/       \_/       \_/       \_/       \_/       \_/       \_/
    |                   |                   |                   |
    |                   |                   |                   |
    |_________          |_________          |_________          |
    |         |         |         |         |         |         |
    |         |         |         |         |         |         |
    |         |         |         |         |         |         |
____v____ ____v____ ____v____ ____v____ ____v____ ____v____ ____v____
\AU-SUBS/ \AU-MONT/ \ US-DB / \ US-JOB/ \QS-HELP/ \QS-MENU/ \ NQ-JCL/
 \-DTL /   \H-DTL/   \-DTL /   \-DTL /   \-DTL /   \-DTL /   \-DTL /
  \___/     \___/     \___/     \___/     \___/     \___/     \___/
```

Figure 1: ROOTDB IMAGE/3000 Data Base

The sections which follow relate each part of the ROOTDB structure to the functions performed by NUCLEUS.

AUDIT INFORMATION MANAGEMENT

The following are needs commonly faced by the DP Manager with respect to the management of audit information:

1. Maintenance of organization-wide financial audit controls;
2. Management of full access passwords to data bases;
3. Timely attention to corrupted data structures;
4. Nonstandard user access controls; and,
5. Immediate user notification of corrupted data structures.

Using that part of the ROOTDB structure shown in Figure 2, NUCLEUS provides a centralized approach to these needs.

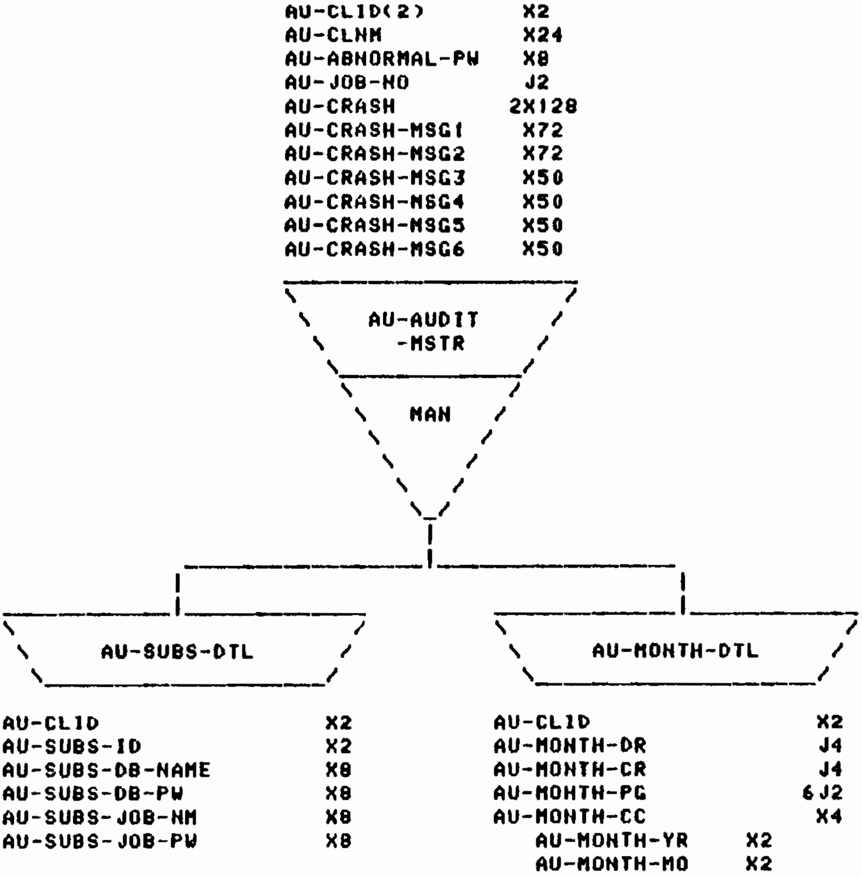

AU-CLID(2) X2
AU-CLNM X24
AU-ABNORMAL-PW X8
AU-JOB-NO J2
AU-CRASH 2X128
AU-CRASH-MSG1 X72
AU-CRASH-MSG2 X72
AU-CRASH-MSG3 X50
AU-CRASH-MSG4 X50
AU-CRASH-MSG5 X50
AU-CRASH-MSG6 X50
AU-AUDIT -MSTR
MAN
AU-SUBS-DTL
AU-MONTH-DTL
AU-CLID X2
AU-SUBS-ID X2
AU-SUBS-DB-NAME X8
AU-SUBS-DB-PW X8
AU-SUBS-JOB-NM X8
AU-SUBS-JOB-PW X8
AU-CLID X2
AU-MONTH-DR J4
AU-MONTH-CR J4
AU-MOHTH-PG 6J2
AU-MONTH-CC X4
AU-MONTH-YR X2
AU-MONTH-MO X2

Figure 2: Audit Information Data Sets

The master data set, AU-AUDIT-MSTR, stores the client's name, an Abnormal Usage Password and system crash results. When the user logs onto the system (in NULOGON), the program determines if the user is logged onto an LDEV which is not assigned to that user or if the user is logging onto the system on a day or time which is not defined for normal usage by the user. If this occurs, the user is prompted to supply the Abnormal Usage Password. Failure to provide the Abnormal Usage Password results in the user being automatically logged off the system. The Abnormal Usage Password should be changed frequently.

If an error has been detected, the program detecting the error condition can call a NUCLEUS supplied routine, NURLCRASH, which will post descriptive information about the error in AU-AUDIT-MSTR. This descriptive information includes the name of the program; the location in the program where the error was detected; the values of the control structure variables; and, a description of the error along with remedy advice. If present, crash information is displayed to the user at log-on time along with a caution not to continue using the system until the remedy advice has been acted upon.

This method of crash management provides assurance that all users are notified of a potential problem; and more importantly, assures that the maintenance effort is directed toward the cause of structure corruption rather than toward problems expressed by subsequent users of a corrupted structure.

Subsystem detail is held in AU-SUBS-DTL. For each subsystem installed, uniquely identified by a two character code, the full access password for associated data bases is maintained along with the MPE user name and password. The MPE user name and password is used for the launching of batch jobs. For example, the entry associated with NUCLEUS is NU for the subsystem ID; ROOTDB for the data base name; ZIPPER for the full access password to ROOTDB; NUBATCH as the batch job user; and, NUBTCHPW as the batch user password.

A NUCLEUS supplied subroutine, NURLGTDBAPW, is called programmatically to allow programs to gain access to data bases with full access capability. This scheme provides the DP Manager with the much needed capability to centralize the management of data base passwords. By simplifying the management of these passwords under one structure, data base passwords can be changed at frequent intervals without disrupting existing in-house software.

The user dialogue manager, NURUNJOB, always asks as its first question, "To which system do you desire access?" A menu of two character codes with associated descriptions follows the prompt. The two character code is used to access AU-SUBS-DTL. All user requests to launch a job will result in the use of the retrieved batch job MPE user name and password for the STREAM job. As with data base passwords, this scheme centralizes the security and management of user initiated batch jobs.

Selected audit information is maintained in AU-MONTH-DTL; one entry is made for each month. An entry contains pages printed by part; checks printed; pages printed to COM; and, monthly total debits and credits. A NUCLEUS supplied routine, NURLEXIT, is called programmatically to post this information to ROOTDB. Maintenance of this information is useful in ordering paper stock and, in addition, provides an important audit control for financial subsystems.

USER MANAGEMENT

The DP Manager needs a simple way in which to manage users. Specifically, there is a need to manage three types of information: General log-on access information; data base access capabilities at the data base, data set and data entry level; and, interactive and batch job running capabilities. NUCLEUS provides a simple means whereby the DP Manager may insure that users are properly coordinated. User management information is maintained in that part of the ROOTDB structure shown in Figure 3.

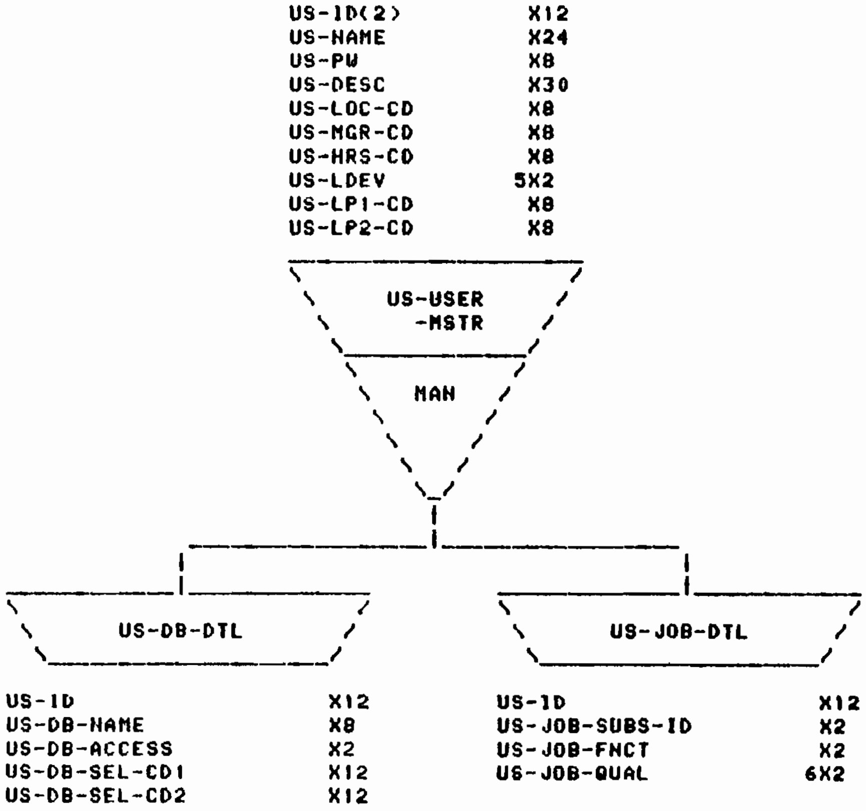

US-ID(2) X12
US-NAME X24
US-PW X8
US-DESC X30
US-LOC-CD X8
US-MGR-CD X8
US-HRS-CD X8
US-LDEV 5X2
US-LP1-CD X8
US-LP2-CD X8
US-USER -MSTR
MAN
US-DB-DTL
US-JOB-DTL
US-ID X12
US-DB-NAME X8
US-DB-ACCESS X2
US-DB-SEL-CD1 X12
US-DB-SEL-CD2 X12
US-ID X12
US-JOB-SUBS-ID X2
US-JOB-FNCT X2
US-JOB-QUAL 6X2

Figure 3: User Management Information Data Sets

User descriptive information is maintained in US-USER-MSTR. Along with name, description and secondary password, each user has the following information associated:

- A location code used for the distribution of centrally generated reports.
- A manager code which identifies the person authorized to manage the user's data base access and job running capability.
- A code which identifies the day and the hours of the day when the user may be logged onto the system.
- A line printer code which identifies the printer the user prefers to use for standard reports.
- A line printer code which identifies the printer the user prefers to use for the printing of special forms.

When the user logs onto the system, NULOGON creates a temporary file, USERID, which contains all the user's information from US-USER-MSTR. During log-on, the user is given the opportunity to alter line printer assignments. These line printer codes are subsequently used by NURUNJOB for the creation of line printer file equations in JCL.

User access to data base information is managed in US-DB-DTL. Each entry in this data set contains the name of the data base; the name of the data set within the data base; an access code which allows READ, WRITE, conditional or NO access capability; and, two selection codes which are treated as a range.

The NUCLEUS supplied routine, NURLCANYOU, is called programmatically to return the least restrictive access capability found in US-DB-DTL. This technique provides the important capability of easily restricting user access to particular entries in a data set. For example, if a particular program provides interactive access to a General Ledger activity data set, entries could be made in US-DB-DTL which would restrict user access to transactional activity for particular account numbers identified in the selection codes.

Job running capabilities for each user are maintained in US-JOB-DTL. Jobs are initiated by NURUNJOB to be run interactively or launched. Before a requested job is run, NURUNJOB examines US-JOB-DTL to determine if the user is authorized to run the job. Information maintained in US-JOB-DTL identifies the subsystem; the function within the subsystem; and, up to six additional qualifications which further define jobs the user may request. After first asking the "Which system...." question, NURUNJOB asks, "Which function in the .... system would you like to perform?" The two character codes selected from the menues presented for these first two questions plus all other menued responses make up the key which is used by NURUNJOB to scan US-JOB-DTL; i.e., a job request is represented in the system as a progressively concatenated set of coded replies to menu-driven NURUNJOB dialogue. By using the "wild card" for any two character codes in US-JOB-DTL, it can be seen that user job running capabilities can easily be specified as very broad or very narrow.

This technique of user job capability management provides a very convenient centralized way of addressing the troublesome "who can do what" dilemma faced by the DP Manager. In fact, this allows the DP Manager to delegate user management to selected end users; user managers may not assign data base access capabilities or job running capabilities beyond those assigned to themselves.

THE USER INTERFACE

Perhaps the most challenging problem faced by the DP Manager is the implementation of systems which are consistently, simply and kindly interfaced to the user. This requires a scheme for the management of user-system dialogue; a simple means of translating user requests into launched batch jobs or activated processes; and, the creation of a user environment which will be protective and secure, while at the same time not restrictive. Using the part of the ROOTDB structure shown in Figure 4, NUCLEUS provides an effective means of defining the 'user interface'.

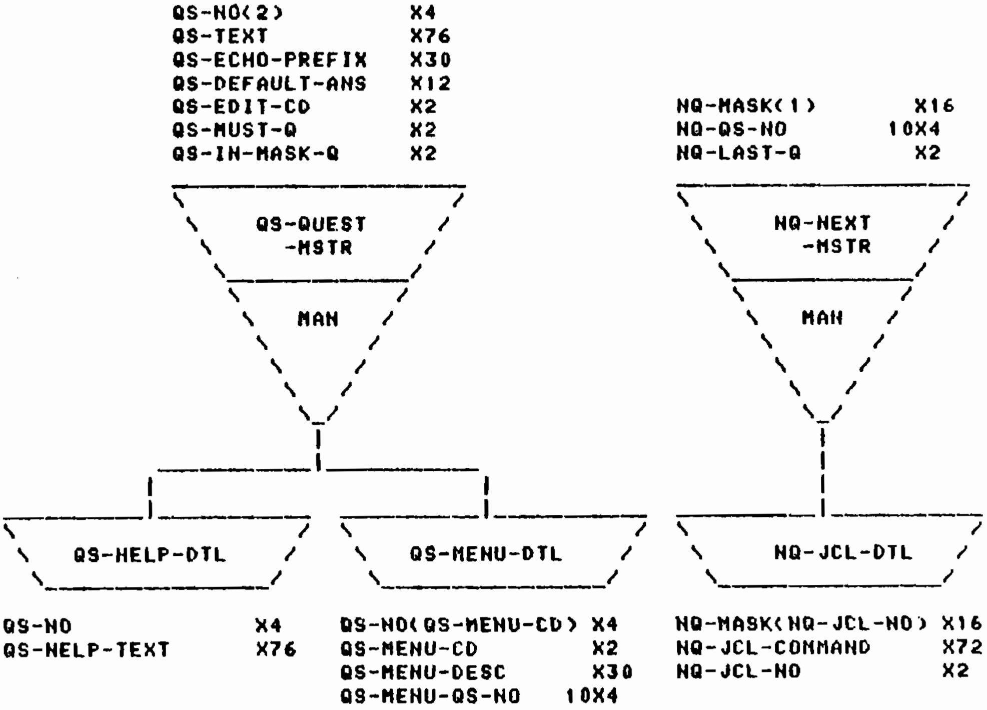

QS-NO(2) X4
QS-TEXT X76
QS-ECHO-PREFIX X30
QS-DEFAULT-ANS X12
QS-EDIT-CD X2
QS-MUST-Q X2
QS-IN-MASK-Q X2
QS-QUEST -MSTR
MAN
NQ-MASK(1) X16
NQ-QS-NO 10X4
NQ-LAST-Q X2
NQ-NEXT -MSTR
MAN
QS-HELP-DTL
QS-MENU-DTL
NQ-JCL-DTL
QS-NO X4
QS-HELP-TEXT X76
QS-NO(QS-MENU-CD) X4
QS-MENU-CD X2
QS-MENU-DESC X30
QS-MENU-QS-NO 10X4
NQ-MASK(NQ-JCL-NO) X16
NQ-JCL-COMMAND X72
NQ-JCL-NO X2

Figure 4: User Interface Data Sets

User dialogue can be represented as a directed graph, where each node represents a question; each arc represents a possible response; and, the terminal node represents the running of a job. Thus, in a theoretic perspective, a job is defined as a set of job control statements which are uniquely identified by a key made up of progressively concatenated user responses to job request dialogue.

To implement this theoretic concept, NUCLEUS manages the directed graph in NQ-NEXT-MSTR and manages dialogue text (questions, answers and HELP text) in QS-QUEST-MSTR. Each entry in NQ-NEXT-MSTR contains a mask of selected progressively concatenated coded user dialogue reponses; a list of up to ten questions which are to be asked in the formation of a new mask; and, a flag which indicates if the mask represents a terminal node. Terminal node masks contain JCL in NQ-JCL-DTL which are used for running the selected job.

As an example of the use of this structure, consider the dialogue involved in a request for a file listing of user information from NUCLEUS. Beginning with a null mask, NURUNJOB asks one question before forming a new mask; "To which system do you desire access?" The user replies "NU" to designate the NUCLEUS system. NURUNJOB uses "NU" to locate an entry in NQ-NEXT-MSTR. This entry represents a nonterminal node which designates one question to be asked before the formation of a new mask. The designated question asks, "Which function in the NUCLEUS system would you like to perform?" The user replies from the menu presented "FL" (ROOTDB File Listing). NURUNJOB uses this new mask "NUFL" to locate the next entry in NQ-NEXT-MSTR.

The located entry, identified as a nonterminal node, designates two questions to be asked in the formation of a new mask. The first question asks, "Which information in the ROOTDB would you like listed?" The user responds from the menu presented "US" (User Information). The second question asks, "How many copies of the listing would you like?" This second question is identified in QS-QUEST-MSTR as an incidental question which doesn't participate in the formation of a new mask. Thus, NURUNJOB uses a mask of "NUFLUS" to locate the next entry in NQ-NEXT-MSTR.

The located entry identified as a terminal node designates one question to be asked before running a job. The question to be asked is, "Which part of the User Information would you like listed?" The user responds "AL" (All User Information). After asking this final question, NURUNJOB uses the setting of the terminal node flag to determine that the job is to be launched. The JCL to be used for launching the job is found in NQ-JCL-DTL.

The first part of each job streamed by NURUNJOB contains a RUN command for the NUCLEUS supplied program NURUNREQ, followed by the identifying number of each question asked and each user response given during the NURUNJOB dialogue along with all user descriptive information contained in USERID. NURUNREQ reads this information from the STREAM file and creates two job temporary files: USERID and RUNREQ. USERID contains user descriptive information and RUNREQ contains user job launch dialogue information. NURUNREQ can then be used by application programs as a general parameter interface. USERID provides useful information to the application program and necessary information to several NUCLEUS supplied routines such as NURLCANYOU and NURLCRASH.

When user dialogue results in a request for an interactive job, NURUNJOB issues all nonRUN MPE commands through the COMMAND intrinsic and initiates execution of all RUN commands by creating and activating the named program as a SON process. When the activated task runs to completion or terminates abnormally, control is returned to NURUNJOB where the next entry in NQ-NEXT-JCL is processed. When an "End of Chain" is detected in NQ-JCL-DTL, NURUNJOB returns to the first user dialogue question, "Which system ....?"

NQ-JCL-DTL is a simple structure where each entry contains a sequence number and a command. Several special commands are supported by NUCLEUS; however, discussion of these details is beyond the scope of this paper.

QS-QUEST-MSTR contains the text along with additional qualifications for each question that may be asked during user dialogue. Additional qualifications for each question are described below:

- An echo prefix used to abbreviate the question during a NURUNJOB prompt which echoes each question asked along with the user response for user verification immediately prior to the launching of a job.

- A default answer value which is used if the user presses the RETURN key without entering a response.

- An edit code which specifies the type of verification to be performed on the user response.

- A code which indicates if the user response is to be concatenated onto previous user responses in the formation of the mask used to access entries in NQ-NEXT-MSTR.

- An indicator as to whether the question is a "MUST" question when processing previously catalogued user responses. NUCLEUS includes a feature which allows the user to catalogue responses given during a NURUNJOB session. Thus, the user can create a file of responses which will result in the activation of one or more jobs in sequence. This feature is a significant timesaver for those users who consistently request the same jobs and would like to use previously catalogued responses placed in a file in much the same way as a UDC. When NURUNJOB is accepting responses from a previously created catalog file, there are certain questions which "MUST" be asked of the user interactively. For example, a question requesting a report data would likely be identified in QS-QUEST-MSTR as a "MUST" question.

Questions in QS-QUEST-MSTR which require a menu of possible responses will be linked to those responses in QS-MENU-DTL. With the question number as the key, each entry in this data set contains the following data items:

- A two character code which can be entered by the user to select a particular entry in the menu.
- A 30 character description of the menu code.
- Up to ten questions may be forced if a given menu code is selected. There are certain questions which should only be asked if the user selects a particular code from the menu. These questions tend to be incidental in nature and their inclusion in the question numbers contained in NQ-NEXT-MSTR would be inconvenient. For example, if the question asked is, "How many copies the user wants of a particular report?" & if the user replied "Two copies", you may wish to force the question, "Are those to be original copies?" at that point.

QS-HELP-DTL allows the system designer to associate free form text with each question. This text is displayed by NURUNJOB anytime the user responds "HELP" to a question asked.

The user dialogue management performed by NUCLEUS has been briefly described in this section. The system designer will find this aspect of NUCLEUS to be extremely easy to use and provides a structure regularity to the user interface. Experience has shown that user acceptance of developed systems is, in large part, a function of the ease and friendliness in which user needs are translated into tasks performed.

COMMON CODE INFORMATION

Many installations are faced with coded information management problems. Specifically:

1. Different encoding structures tend to be developed with each applicational system.
2. The same information may be encoded differently by separate systems.
3. Disparate, separately designed structures do not lend themselves to centralized coordination of code maintenance.
4. Program development and maintenance are more difficult when dealing with a variety of code structures.
5. Multiple, separately maintained code structures make it very difficult to insure that users have hard copy documentation which lists current code assignment values.
6. Many applicational systems do not include a facility for user interactive inquiry into current code assignment values.

Using the part of the ROOTDB structure shown in Figure 5, NUCLEUS provides a means of adaquately addressing these problems.

```
CD-CC(0)           X12
   CD-CATEGORY X4
   CD-CODE     X8
CD-DESCS           X8
CD-DESCM           X30
CD-DESCL           X72
CD-ASSOC-VAL     5J4
CD-ASSOC-CODE    5X8
CD-ASSOCI-DESC   5X30

_____________________
\                   /
 \    CD-CODES     /
  \    -MSTR      /
   \_____________/
    \           /
     \   MAN   /
      \       /
       \     /
        \   /
         \_/
```

Figure 5: Common Codes Data Set

For each code contained in CD-CODES-MSTR, the following data items may be associated:

- A four character code category. For example, "PYAA" could stand for a Payroll Affirmative Action codes category.
- An eight character code value.
- A short (8 character), medium (30 character) and long (72 character) description of the code value.
- Up to five, double long integers, handled internally like COBOL PIC S9(13)V9(5) variables.
- Up to five, eight character associated code values.
- Up to five, 30 character associated code value descriptions.

This structure tends to be very inefficient with respect to disc utilization; however, most installations will find that the centralization of encoded structures will often result in a net sector savings.

HEADING INFORMATION

The movement of report headings and screen headings from programmatic definition to a centrally managed structure, HD-HEADS-MSTR, provides several advantages:

1. Heading standards may be applied consistently.
2. The programming task is made easier.
3. A convenient structure from which to answer the question, "What reports do we currently support?"
4. Pagination, titles and general trivial format changes can be made without recompiling programs.

Heading information is maintained in that part of the ROOTDB structure shown in Figure 6.

HD-ID(0)	X8
HD-TITLE	X50
HD-TITLE-DT	X2
HD-TITLE-SEL	X2
HD-FORMAT	X2
HD-COMPRESS	J
HD-TOP-SPACE	J
HD-DETAIL-SPACE	J
HD-LENGTH	J
HD-WIDTH	J
HD-CLOSE	J
HD-HEAD	6X132

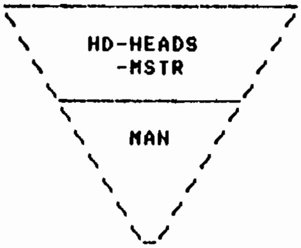


Figure 6: Heading Data Set

For each entry in HD-HEADS-MSTR, up to six heading lines are maintained along with a title and other format directive qualifiers. The format directive details are not described in this section.

ERROR INFORMATION

Most DP environments are troubled by the following error management problems:

1. Complaints by users that error messages are unclear.
2. Complaints by users and systems' staff that the proper course of action when an error occurs is undefined.
3. Due to disparate programmatically defined errors, it is difficult for the manager to obtain or provide current listing of all errors identified by software.
4. Most errors detected do not include a qualification of the level of their severity.

Error information is maintained in that part of the ROOTDB structure shown in Figure 7:

```
ER-CC            X8
   ER-ID      X6
   ER-NO      J
ER-DESC          X72
ER-NATURE        X2
ER-ADVICE        X72

_____________________
\                   /
 \    ER-ERROR     /
  \     -MSTR     /
   \_____________/
    \           /
     \   MAN   /
      \       /
       \     /
        \   /
         \_/
```

Figure 7: Error Data Set

ER-ERROR-MSTR is designed to address the above stated problems. Each error is qualified by a six character Subsystem ID code; a four digit error number which is unique within each Subsystem ID; a 72 character error description; a two character indicator of the level of severity of the error; and, a 72 character description of what should be done to correct the error.

SUMMARY

Describing a major system such as NUCLEUS in a limited Proceedings article leaves one erring on the side of brevity. It is hoped that this paper has fairly captured the conceptual framework of NUCLEUS as well as to describe some of the major components. Simply stated, NUCLEUS is a data structure which addresses many of the more pressing DP management needs in a multi-user HP3000 applicational environment. Most of the system is written in COBOL, with all I/O and support subroutines written in SPL.

NUCLEUS includes all software necessary to maintain the ROOTDB data base. All updates to the data base are done interactively and use VIEW forms. All software for the generation of file listings from the data base is also included, along with software which provides interactive inquiry into any part of the data base. In addition to these functions, NULOGON, NURUNJOB and NURUNREQ are included along with several support subroutines.

ITA / 3000

A User-Friendly Screen Form Handler operating in Character Mode

by

Kurt Sager
SWS SoftWare Systems AG
Schoenauweg 8
CH-3007 Bern / Switzerland
phone +4131 46 16 64
telex 33388 sws ch

'1983 HP3000 International Conference Edinburgh'
Scotland, 2nd-7th October 1983.

Contents

Summary

1. Overview of programming techniques for dialogs between terminal users and application programs running on the HP/3000 system

2. Requirements for user-friendly dialogs on display terminals

   2.1 Forms design and EDP standards

   2.2 Interaction between terminal user and computer program

   2.3 Run-time efficiency

3. Features of the ITA/3000 Character Mode Screen Form Handler

References

Summary

Most online application systems interact with the terminal user by means of screen forms. The features of the screen form handler used in on-line programs contribute in a significant way to the user-friendliness of an online system therefore.

ITA/3000 screen form handler has been designed and developed with the user in mind:

- very fast visual forms design and forms changes using any editor,
- demonstration to the future end-user by a check and demo utility,
- definition of system- or account-wide standards for various run time features, such as field enhancements,
- highlightening of the field where the cursor actually is,
- all input terminated by the RETURN key
- user input is immediately checked for formal correctness, and further verification accessing any other data if necessary,
- application program may read one, several, or all fields in a form,
- if invalid data is entered an error message is displayed in the users local language, then cursor is repositioned at the invalid field,
- instantanous forms change by using local terminal memory which is part of every HP terminal,
- fast response time thanks to character mode operation and small stack requirements.

ITA/3000 is available as a fully supported vendor product.

1. Overview on programming techniques for dialogs between terminal users and application programs running on the HP/3000 system

The first technique used since the early days of interactive computing is the so-called 'question / answer sequence' where the terminal user has to answer a question displayed (or a prompt sign, meaning that the program is waiting for the next user command) before the program continues. It remains the prefered method in many situations, e.g. for interacting with the operating system MPE, in utility programs, line editors, and most scientific programs with few input options and a lot of calculations.

For commercial data processing however, the today's standard method of interaction between a terminal user and a well designed computer program is the 'show a form & fill-in the blanks' technique.

It has become more and more used for on-line data processing applications and is

- appropriate for well defined and often used applications,
- the typical dialog technique on display terminals for commercial applications,
- similar to traditional paper work,
- well-suited for unskilled people and occasional users,
- easy to see the relations between many fields to be filled in, e.g. dozens of descriptive and quantitative inputs for a new stock item to be entered.

Using this method, an entire form is displayed on the terminal screen, typically showing a title, some explanations, a selection of options, or questions followed by distinctly enhanced fields for the user input. The cursor is positioned at the beginning of an input field, ususally the first one. The user has time to read the instructions on the screen, to understand all the questions to be answered. (Generally it helps a lot to see all related questions on the same page.) The user then answers the first question. If the first input field is completly filled, or if a defined 'termination key' is pressed, the cursor jumps automatically to the beginning of the next input field. If the user types a formally or logically invalid answer, an error message explaining the problem is displayed as soon as possible, and the user is asked to correct his input.

The 'Show a form & fill in the blanks' method has the following main advantages over the older 'question / answer technique':

- It looks more professional to show entire forms on the terminal screen and to let the user fill in the blanks.
- It is easier for the terminal user to know what kind of input is expected.
- Possibly the user has the option to check his input in all fields by re-reading his input, and to correct any bad values.

- Forms are displayed at once and then remain motionless on the screen until an other form is needed. This is much more agreeable to the operators eye's than the frequent jumping up of lines as with the 'question / answer sequence' technique.

Data transmission between a terminal and a HP3000 computer occurs in character mode or in pseudo block mode, depending on the terminal type and the screen form handler used:

* VPLUS (by Hewlett-Packard) works in pseudo block mode and needs block mode terminals (2640B, 2645, 2622 ff).

* ITA/3000 (by SWS SoftWare Systems [3]), works in pure character mode with every HP display terminal, including the low-cost 2621.

The technical differences of the two transmission modes and the impact they have on system load and programmer needs has been documented in the two recent articles [1], [2].

2. Requirements for user-friendly dialogs on display terminals

Everybody agrees today that display terminals should satisfy basic ergonomic requirements relative to the following hardware aspects:

- keyboard design: flat, easily reachable position of special keys, separate numeric group, agreable typing
- display quality: flicker-free and easy-to-read, adjustable brightness and focussing, choice of display colors,
- display unit: twistable, to accomodate for external lighting conditions.

Hewlett-Packard now offers several optional features for the 262x line terminals, and on a future terminal line many new ergonometric insights will certainly be realised.

But what about 'ergonomics' in the software domain, for example

- easy to understand screen forms, really reflecting the user needs, similar to previous paper work?
- system feedback as the user fills in every field?
- response time to allow smooth and continuous data entry operations?

Critical inspection of existing on-line applications and discussions with (satisfied?) terminal users often emphasizes the following problems:

- Depending on the designer's style the form lay-out may look quite different from program to program: same field types (display-only, required, input, ...) are enhanced in different ways, error handling is implemented by mysterious error numbers or messages hardcoded in the program code and difficult to modify or to translate to another language.

- End-users rarely have seen the screen form before the first production run, except possibly as part of system specifications on paper drawings. How many users really interact with the system analyst at forms design time? Not only at the conference table but seeing the forms on the screen, typing data in the input fields and jumping from form to form (that's the first step in prototyping a new application).
- On more complex forms having many dozens of input fields the actual cursor position is difficult to locate visually if the current field is not distinctly enhanced.
- It is confusing to the terminal user if type checking is not done immediately. But can a dp manager justify the extra expense of nearly $1000 for every 2624 display (compared to the cheaper 2622) just to have immediate type checking?

Many of the problems above cannot easily by avoided if HP's VPLUS software is used to implement the 'show a form & fill in the blanks' technique.

But the character mode screen handler ITA/3000 has been designed with the user in mind:

- very fast visual forms design and forms changes using any editor,
- demonstration to the future end-user by a check and demo utility,
- definition of system- or account-wide standards for various run time features, such as field enhancements,
- highlighting of the field where the cursor actually is,
- all input terminated by the RETURN key, e.g. no error-prone switching between RETURN, TAB, and ENTER key,
- user input is immediately checked for formal correctness, and further verification accessing any other data if necessary,
- application program may read one, several, or all fields in a form
- if invalid data is entered an error message is displayed in the users local language, then cursor is repositioned at the invalid field,
- instantanous forms change by using local terminal memory on every HP terminal, even the low-cost HP2621,
- fast response time thanks to character mode operation and small stack requirements.

2.1 Forms Design and EDP standards

ITA/3000 forms are designed with the aid of your preferred editor: QEDIT from Robelle Consulting, TDP or EDITOR from HP, then kept as standard ASCII or as compact QEDIT file:

Using editor commands you visually design the desired form as follows:

- Write any constant text such as title, explanations, user directives, and field descriptions at the desired locations. Full screen editing may be helpful to do this design task.
- Delimit every field which contains variable data at run time by braces { } if it is a display-only field, or by brackets [ ] if it is an input/output field.

  On national terminals (german, french, spanish, scandinavian, ...) where the symbols {}[] are replaced by national characters the alternative symbols ( ) for display-only fields, and < > for input/output fields can be used.
- Inside the field delimiters {},[] (or (),<> on national terminals) you specify the desired data type by filling them with one of the field type symbols shown in the table below.

Field Type	Checking	Conversion to internal types		
		IMAGE	COBOL	SPL
[ ] 1..n	no checking, alphanumeric	Xn	pic X(n)	byte array
[UUUU]	upshift lowercase alphabet.	Un	pic X(n)	byte array
[****]	no-echoing (e.g. passwords)	Xn	pic X(n)	byte array
[ZZZZ]	numeric	Zn	pic 9(n)	(byte array)
[9999]	numeric, up to 4 digits	J1	pic 9(4) comp	integer
[99999]	numeric, 5 to 9 digits	J2	pic 9(5) comp	double
[PPPPP]	numeric	P6	pic 9(5) comp-3	n/a
[DD.MM.YY]	date	X8	pic X(8)	byte array
[DD/MM/YY]	date (--> YYMMDD)	Z6	pic 9(6)	(byte array)
[DD MM YY] [DD MM YYYY]	date (--> YYYYMMDD)	J2	pic 9(8) comp	double
[YY.WW]	year & week	J1	pic 9(4) comp	integer
[YY WW]	year&week (-->YYYYMMDD)	J2	pic 9(8) comp	double

All numeric field types (symbols 9, Z, or P) may optionally contain a decimal point, or a leading 'minus sign' to allow for negative values.

When the forms for an application are drawn, it may be checked and demonstrated to the end-user immediately by using the ITACHECK utility program. Before any program coding is done the form(s) appearance and jumping from form to form can be demonstrated. Field enhancements and other attributes can be changed as long as not fixed by account or system manager. The end-user can try out the forms by entering data and using function keys.

If any form design or field attributes such as type or length do not satisfy the user needs, they can be changed using the editor in a matter of seconds, then redemonstrated again. Compare this fast developement cycle with the VPLUS approach: lengthy forms design procedure through FORMSPEC, then very time-consuming forms compilation!

EDP standards can be recommended or enforced by the account or system manager by setting optional or required forms attributes using the ITASET utility. The programmer can overwrite the optional attributes only from his application program (if necessary). This ensures that in a given account or for the whole systems all the ITA/3000 forms attributes can be controlled for the end-user's benefits.

2.2 Interaction between terminal user and computer program

The user-friendliness of the screen form handler ITA/3000 is determined by the way it interacts with the terminal user.

Special Enhancement of Current Field: Especially in complex forms with many input fields the operators eye's get tired while looking for the actual position of the blinking cursor (underline sign) which marks the current input position. ITA/3000 allows to choice a special field enhancement for the current input field, for example 'full bright inverse video' if the normal input field enhancement is 'half bright inverse video'. Using is unique feature the terminal user can locate the current input field instantaneously.

Using RETURN for all input termination: If the Block Mode technique is used (as with VPLUS) then you have to press 'RETURN' after the HELLO and the :RUN command, then to use the 'TAB' key to terminate every input in a screen field, and finally to press the 'ENTER' key which sends the whole bunch of data to the computer. With ITA/3000 the RETURN key is used in all cases to terminate an input operation.

Immediate input checking: The user feels much more comfortable with his answers to the computer's questions if his input is immediately checked at least for formal correctness (alphabetic/numeric/number of decimals ...) as this is done by ITA/3000. If the data entered does not follow the field type specifications then just after the RETURN key is hit an error message is displayed an the cursor is repositioned at the beginning of the field in error. Except for the more expensive HP2624B display terminal VPLUS operating in Block Mode can do such simple type checking only after the ENTER key has been pressed. This may be minutes later when the user has finished entering many other fields on the screen.

If the data entered needs access to other data in the computer for verification, then ITA/3000 can be directed to read just this one (or a few) fields. The program may then access other data (i.e. the customer record in a data base if the customer number was typed-in). Optionally it can display additional information or a specific message if the check fails. It then asks for the next user input by positioning the cursor to the appropriate field according to the action to be taken.

2.3 Run-time efficiency

Good response times are of major concern to the terminal user. Indeed it is frustrating and a waste of time (equals money!) to sit behind the screen and waiting until the computer is ready to accept the next input.

The following rules help you to design interactive programs with high run-time efficiency:

* Read Robert Green's paper on 'Optimizing On-line Programs' (see reference [4]).

* Make use of local terminal memory to hold the most often called screen forms:

  - If your programs call ITA/3000 then on every HP display terminal (including the 2621) the most frequently used forms are reactivated instantaneously and automatically.

  - If you insist to use VPLUS then you need the expensive 2624 or 2626 terminals.

* Only the data you enter on the keyboard should be transmitted to the computer:

  - That, by definition, is what ITA/3000 does while operating in character mode!

  - The contents of all unprotected fields are sent to the computer at maximum speed (e.g. 9600 baud) every time you hit the ENTER key in a VPLUS form, even if you changed just a few bytes in the current screen. Using form families the amount of data transferred can be reduced for the most frequent situations, but at the expense of more forms design and programming time.

* Keep the code and data stack size as small as possible:

  - Minimize usage of global data: ITA/3000 just needs about 100 words for an average form with 20 fields. Compare this small amount with the several thousend bytes VPLUS needs in the DB-DL area of the data stack!

  - Minimize code segment size: do not put relatively seldom used error messages in the code segments, but to MPE catalog files. ITA/3000 uses error catalog files in English, French or German to hold all its messages as well as the most frequent messages from the file system and from IMAGE. You may expand this catalog by your own application specific messages.

3. Features of the ITA/3000 Character Mode Screen Form Handler

The following table lists the most important features of ITA/3000 taking HP's VPLUS as reference:

Features	ITA/3000	V/3000
* transmission mode (terminal to computer)	character	block
* immediate input checking	yes	2624 only
* error messages in local language	yes	no
* instantanous forms change	yes	2624&2626
* number of forms resident in local memory	2	1, more on 2624&2626
* no-echo fields for passwords	yes	2624 only
* automatic & immediate right justification of numeric data	yes	2624 only
* softkeys supported	yes	yes
* time-out feature	yes	yes
* special 'current field' enhancement	yes	no
* system/account wide defaults & standards	yes	no
* 'visual' forms design by	QEDIT/EDITOR	FORMSPEC
* maximum number of fields per form	unlimited	128
* number of procedures to remember	1	24
* forms compilation	none	time consuming
* field processing specifications	progr. only	forms & progr.
* automatic type conversion	yes	no
* runs on low cost terminal HP2621	yes	no
* test input optionally from disc file	yes	no
* mixing forms and standard writes	easy	difficult
* program development time	short	longer
* automatic buffer declaration (COBOL,SPL)	yes	no

References

[1] John Hulme, The Block-Mode Conspiracy - A Case of Ignoring user Input?, Proceedings 1983 HP3000 International Conference Montreal April 24-29

[2] Kurt Sager and Eddie Stiefel, Character versus Block Mode Terminal Input on the HP3000, SUPERGROUP Newsletter, Vol.2, Issue 4, October 1982

[3] ITA/3000 Reference Manual by SWS SoftWare Systems AG, Schoenauweg 8, CH-3007 Berne, Switzerland

[4] R.M. Green, Optimizing On-line Programs, Technical Report, 1981, Robelle Consulting Ltd. (see p. 49)

[5] A.R. Morris, V/3000 failed at the CCSSRD, Newsletter HP Bonneville Regional Users Group, 1982

IMAGE and ADAGER: The dynamic duo.

F. Alfredo Rego and Fred White

Adager
Apartado 248
Antigua
Guatemala

Telephone (502-2) 324333
Telex 4192 Teltro Gu

IMAGE/3000 is the database management system (DBMS) for the family of Hewlett-Packard HP3000 computers. ADAGER is the database-utility software system that complements and enhances IMAGE.

IMAGE is a software product of Hewlett-Packard Co. ADAGER is a software product of Adager S.A. Both systems fit like hand-in-glove and are fully supported by cooperating research laboratories and field engineers in response to the needs of HP3000 users throughout the world:

- IMAGE is licensed as part of Hewlett-Packard's MPE/3000 fundamental-operating-system (FOS) and supported according to HP's standard maintenance contract.
- ADAGER is licensed directly by Adager S.A. and supported according to Adager's standard maintenance contract.
- A licensed ADAGER user who has contracted ADAGER's maintenance with Adager S.A. may, in addition, contract ADAGER support directly with Hewlett-Packard.

The majority of the HP3000 computers in the world run applications based on IMAGE. The managers of the most successful installations claim that **if your application depends on IMAGE you must also have** ADAGER. Let us review some of the reasons that support this claim.

Quick review of IMAGE

IMAGE is responsible for:

obtaining access to a database (DBOPEN, protected by a formidable security mechanism that allows controlled access down to the field level);
releasing access to a database (DBCLOSE);
storing data in a database (DBPUT);
retrieving data from a database (DBFIND, DBGET);
modifying data which is already in the database (DBUPDATE); and
deleting data from the database (DBDELETE).

Since IMAGE is a mature and reliable DBMS, it provides several facilities associated with:

self-describing data (DBINFO);
data-independence (various LIST constructs);
concurrency control (DBLOCK, DBUNLOCK);
logging and recovery (DBBEGIN, DBEND, DBMEMO, ILR, DBRECOV);
user control over performance tradeoffs (DBCONTROL); and
reporting of database-access errors (DBERROR, DBEXPLAIN).

IMAGE also contains mechanisms for:

describing the structure of a database (DBSCHEMA);
creating and maintaining the database's performance and logging parameters;
erasing and purging the database (DBUTIL);
backing up and restoring a database (DBSTORE/DBRESTOR); and
unloading and reloading the data from/to a database while allowing a limited variety of database-structure transformations (DBUNLOAD/ DBLOAD).

Several Hewlett-Packard computers can be linked (either locally or remotely) to form computer networks. IMAGE, therefore, allows **remote-database access** controlled by strict security provisions.

All in all, IMAGE has qualities that make it a favorite foundation for the development of computerized applications that require:

* reliability
* concurrent access by many mixed processes, both on-line and batch
* controlled access by means of a comprehensive security matrix
* ease of use
* logging (for auditing and recovery)
* sharing of internal buffer pools
* availability of data-entry systems
* availability of reporting systems
* availability of general database-maintenance systems

However,...

The very fact that IMAGE is as solid and reliable as a reinforced-concrete building makes it difficult to change its structures once they are "set".

Enter ADAGER

ADAGER adds another dimension to IMAGE: Flexibility. ADAGER does a variety of transformations and related maintenance functions on IMAGE databases while preserving the semantic meaning of the user's data. ADAGER does not need to unload/reload the database and it uses high-speed compression and data transfer technologies to minimize database-maintenance time.

ADAGER strictly enforces IMAGE's rules and regulations by means of procedures that monitor, diagnose and repair database structural problems.

ADAGER requires that the user's logon identification be of a level consistent with the database's security and privacy. Only the database's CREATOR, logged in the database's GROUP and ACCOUNT can TRANSFORM the database. Other users can access the database in a browsing, non-transformational mode if they pass ADAGER's security checks (for instance, a non-creator must supply the database's maintenance word for a COPY function). Naturally, users with account manager (AM), system manager (SM) and/or system supervisor (OP) capabilities can access a database in well-defined non-transformational ways (like backing it up, decompiling it, copying it, etc.), according to these capabilities. This means that a system supervisor can back up a database but cannot decompile it, for instance. ADAGER treats capabilities in an inclusive-or fashion (i.e., a system supervisor can also be an account manager and/or a system manager). But ADAGER requires a user to have all the required capabilities in an and fashion before the performance of a given ADAGER function.

ADAGER is responsible for:

Backing up and restoring one or more databases to/from an ADAGER-labeled tape set, using high-speed encrypting, compression and transfer technologies with 16 K-byte tape blocks. It allows you to issue standard MPE STORE/RESTORE commands so that you can take care of all your backup needs within a single ADAGER module.

Creating a database, allowing you to place each dataset in a specific disc class or unit. Naturally, ADAGER also allows you to **move any dataset to a different disc class or unit** if you think that would be advantageous for performance reasons.

Decompiling a database and producing a SCHEMA file which is compatible with DBSCHEMA and practically all HP3000 editors.

Cross-referencing and describing the elements of a database with entity numbers and buffer positions.

Describing, adding, modifying and deleting the provisions for security and privacy that apply to database access according to IMAGE's user classes/levels. Since passwords may have unprintable ASCII characters, ADAGER also reports the octal equivalents of passwords.

Copying a database, optionally assigning a new name to the new copy.

Renaming databases, datasets or data items (and their field references in datasets).

Adding data items, datasets, fields, paths and the sort feature to paths.

Deleting data items, datasets, all entries from a dataset (thereby erasing it), **noncritical fields, paths, the sort feature from paths or entries from given path chains.**

Modifying the characteristics of data items (including items which are referenced as search or sort fields in datasets).

Sliding a data item up or down within the item table.

Changing an automatic master to a manual master or a manual to an automatic.

Sliding a set up or down within the set table.

Changing the capacity of a dataset (master or detail).

Changing the value of the search field in all the entries of a path chain (master chainhead and detail chain members).

Repacking a detail dataset according to its primary path (optionally redefining the primary path for a detail with more than one path).

Repacking a master dataset according to its synonym chains.

Reshuffling the fields within a dataset.

Redefining the blocking factor of a dataset.

Cremating a database (i.e., erasing and then purging all the disc space that was assigned to a database, including the root file).

In short:

IMAGE is a powerful and reliable database management system. ADAGER adds the **flexibility** dimension to IMAGE. Both IMAGE and ADAGER, when used together, constitute a very dynamic duo. When they perform on the Hewlett-Packard HP3000 family of computers they provide the most cost-effective foundation for the development of successful database-oriented computer systems.

Have we forgotten something?

If so, you may ask us during the interactive session.

Thank you.

ingenieurbüro jörg grössler gmbh
- edv - beratung & system - unterstützung -

rheinstraße 24
1000 berlin 41
030/852 70 27

Dokumentation

3.83

DATABASE TECHNIQUES - IMAGE VERSA KSAM

AUTHOR: DIPL.-ING.JÖRG GRÖSSLER

DATE: 7/29/83

Berlin Charlottenburg, HRB · Nr. 16966
Geschäftsführer Dipl Ing Jörg Grössler

Database Techniques - IMAGE versa KSAM

Introduction

Since more than 10 years the users of HP 3000 systems have to answer the question of how to store their data in files. Two main subsystems are available. The first one - IMAGE - was introduced around 1975, the other one - KSAM in 1978. The purpose of this paper is to show the philosophy behind these two techniques, to discuss advantages and disadvantages and to give some hints of how to make a decision which database technique is the appropriate one for a specific solution.

What is the basic idea behind IMAGE and KSAM?

For many applications, data has to be stored on disc. Using MPE files, there are two basic methods to organize this data storage:

Sequential Read/Write
Indexed Read/Write

During a sequential read or write, all records are accessed in an ascending order according to their location in the file. Accessing a file by index means that you have to provide a record number which identifies the location of the record in the file.

Usually, data records contain several information items. Each information item can consist of one data type or a set of data types.

As an example, let us consider that a company has to maintain information about 10 employees. The 4 information items to be stored in the file are a personnel number, the last name, the first name and the city, where the individual resides.

PERS No.	Last Name	First Name	City
10	Miller	Nancy	Houston
15	Brighton	Roy	Houston
21	Graziano	Ilene	Houston
17	Brown	Charles	San Antonio
33	Smith	Paula	Beaumont
35	Snider	Jim	Houston
23	Miller	Charles	Houston
27	Benson	George	Brownsville
19	Wood	Michael	San Antonio
36	Landrum	Mary	Houston

Each record contains 64 bytes of information. To store these records on discs, you could create an MPE-file like this:

```
:BUILD PERS;REC=-64,4,F,BINARY,DISC=10
```

Since each individual is identified by the personnel number, which is unique but not necessarily in order, there is a problem, for example, to read the record containing information about Charles Miller. In this case, you would have to provide record number 6 (it is the 7th record, as records are counted starting from 0). This means that a table has to exist somewhere, where personnel numbers and their according record numbers are stored. Than, an access could be performed using the personnel number rather than the record number. It gets even more complicated when you want to access the record by giving the city rather than the personnel number (for example, if you don't know the personnel number). The problem here is, that a given city may appear more than once in the file.

KSAM as well as IMAGE provide a method to access records by key items rather than the record number in the file. In case key items are duplicate, they provide a facility to read all records containing the key item without reading the file in sequential order.

The basic difference between KSAM and IMAGE is the internal way of maintaining the key items.

In principle, it should be unnecessary for the user to know how KSAM and IMAGE internally work (except the user has a basic interest in database techniques). Unfortunately, however, in this case the internal methods of KSAM and IMAGE have a direct effect to their handling, disc space needed and performance.

How does KSAM organize its key items?

Provided that we want to access the data by personnel number and city we have to define two key items.

Building this file is done using KSAMUTIL als follows:

```
>BUILD  PERS;KEYFILE=PERSKEY;REC=64,4,F.BINARY;&
>KEY=B,1,4;KEY=B,45,20,,DUP;DISC=10
```

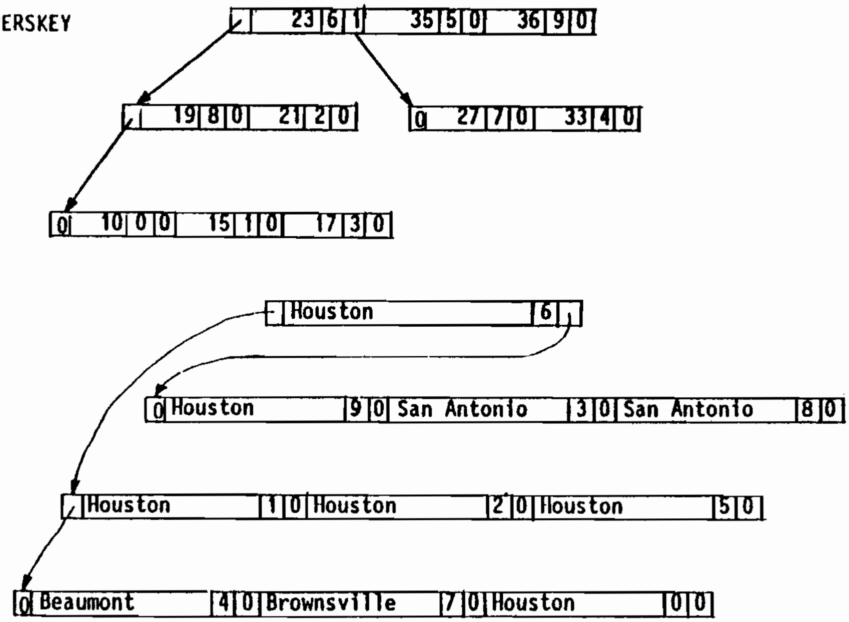
In file PERS, the data is stored without additional information (similar to ordenary MPE-files). Information about key items and the key items themselves are stored in the so called key file PERSKEY. The key information is just given by the location in the record and the length of the item.

And this is (simplified) the structure of the KSAM data and KSAM key file.

PERS

	PERS No.	Last Name	First Name	City
0	10	Miller	Nancy	Houston
1	15	Brighton	Roy	Houston
2	21	Graziano	Ilene	Houston
3	17	Brown	Charles	San Antonio
4	33	Smith	Paula	Beaumont
5	35	Snider	Jim	Houston
6	23	Miller	Charles	Houston
7	27	Benson	George	Brownsville
8	19	Wood	Michael	San Antonio
9	36	Landrum	Mary	Houston

PERSKEY


23 6 1 35 5 0 36 9 0
19 8 0 21 2 0
0 27 7 0 33 4 0
0 10 0 0 15 1 0 17 3 0
Houston 6
0 Houston 9 0 San Antonio 3 0 San Antonio 8 0
Houston 1 0 Houston 2 0 Houston 5 0
0 Beaumont 4 0 Brownsville 7 0 Houston 0 0

In the key file the items are stored in 'key blocks'. The number of items per block can be specified in the BUILD command and is in a way depending on the length of the key item.

This structur of key item storage is known as B-tree. The leaves of the tree are the key blocks. They contain the key items followed by the record number of the data file. There are two pointers for each entry in the key block. One pointer in front of the key points to the part of the tree which contains key items less or equal to the items stored in the block. The pointer after the key item points to the part of the tree which contains key items greater or equal to the key item itself.

When a record is accessed by key, the KSAM system first reads the 'root block' to determine whether the key item is stored in the left part or the right part of the tree. In case the key item is unequal to the items stored in the root block, KSAM reads the next level of leaves to continue comparison. This is done until the key item is found or the lowest level is reached.

It is obvious that the structure of key items in the key file has to be in alphabetic order. Therefore, the user can

- read the data records in alphabetic order,
- access information items which contain not all bytes of the item (for example only the first three bytes).

How does IMAGE internally work?

In case, the same information should be stored in an IMAGE database the user has to define the database structure by using the language of the schema processor:

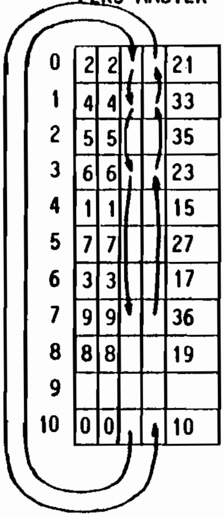
```
BEGIN DATA BASE PERS;

PASSWORDS:      1 PASS;

ITEMS:          PERS-NO,    I1;
                LAST-NAME,  X20;
                FIRST-NAME, X20;
                CITY,       X20;

SETS:

NAME:           PERS-MASTER,M (1/1);

ENTRY:          PERS-NO (1);
CAPACITY:       11;

NAME:           CITY-MASTER,M (1/1);

ENTRY:          CITY (1);

CAPACITY:       11;

NAME:           PERS-SET,D (1/1);

ENTRY:          PERS-NO (PERS-MASTER),
                LAST-NAME,
                FIRST-NAME,
                DITY (CITY-MASTER),

CAPACITY:       11;

END.
```

IMAGE organizes its database in different privileged files. In our case there are the files PERS (the 'root file'), PERS01, PERS02, PERS03 (containing the information of the data sets). The root file contains information about the entire database, for example the names of items and data sets and their capacity.

The data set files contain the information of so called master and detail data sets. The master data sets contain the key information, the detail data sets the data itself.

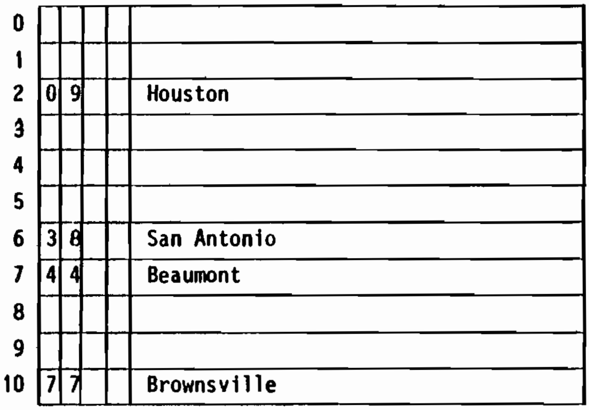
The next picture shows the two master data sets and one detail set defined in our data base PERS.

PERS-MASTER



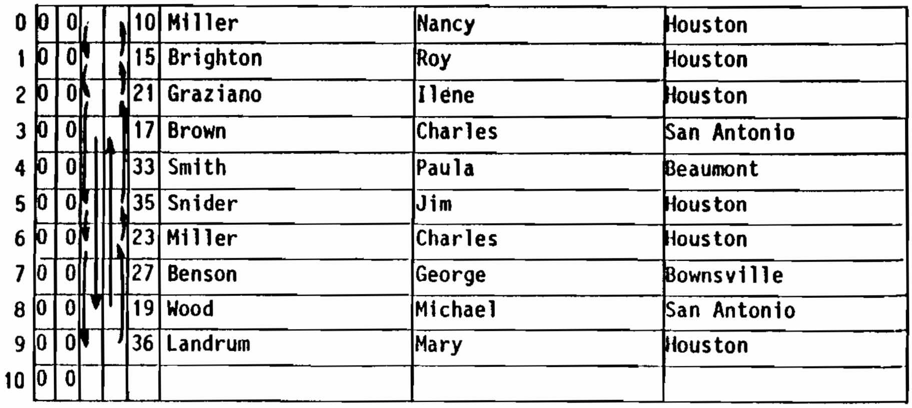
0	2	2			21
1	4	4			33
2	5	5			35
3	6	6			23
4	1	1			15
5	7	7			27
6	3	3			17
7	9	9			36
8	8	8			19
9					
10	0	0			10

CITY-MASTER



0					
1					
2	0	9			Houston
3					
4					
5					
6	3	8			San Antonio
7	4	4			Beaumont
8					
9					
10	7	7			Brownsville

PERS-SET



0	0	0			10	Miller	Nancy	Houston
1	0	0			15	Brighton	Roy	Houston
2	0	0			21	Graziano	Ilene	Houston
3	0	0			17	Brown	Charles	San Antonio
4	0	0			33	Smith	Paula	Beaumont
5	0	0			35	Snider	Jim	Houston
6	0	0			23	Miller	Charles	Houston
7	0	0			27	Benson	George	Bownsville
8	0	0			19	Wood	Michael	San Antonio
9	0	0			36	Landrum	Mary	Houston
10	0	0						

It is essential to know how IMAGE organizes its key items in the master data sets. One data set contains the data item 'Personnel No' the other one the data item 'City'. IMAGE uses a so called hash algorithm to access the master data sets. This hash algorithm works differently for character strings and integers.

In our case, the personnel number is stored as integer. To access an entry in a master data set, a number is calculated to determine the location within the file.

In case the key item is integer, the number calculated is simply a modulo, the capacity of the data set.

In our case the capacity is 11. When the first entry is added to the master data set the location is 10 modulo 11 which is 10. So it gets the 10th location (counted from 0) in the master data set. The next entry is 15 and its location is 15 modulo 11 which is 4. The next entry 21 has in principle the same location in the master data set as the first one (21 modulo 11 is 10). This case is called a 'collision' which IMAGE has to take care of. This means that the next free location is determined which in our case is location 0, starting from the beginning and a so called synonym chain points to the location having the same hash index. The key item 21 is stored in location 0 and location 10 contains a pointer to location 0 and location 0 contains a backward pointer to location 10. In our example, the synonym chains are indicated in column 2 of the master data set 'PERS-MASTER'.

In case the key item is a character string the value is determined by adding all character values together and this value then is taken modulo the capacity of the master data set.

It is characteristic for IMAGE that the data item is stored only once in the master data set even if it occures more than once in the according detail set. The detail set itself contains additional information (in our example colums 1 and 2 of the detail set) where records are linked together having the same master item. In the master data set, two pointers are maintained pointing to the first and last entry of a 'chain' within a detail set having the same master item.

Because of the hash algorithm, the data items in the master data set don't appear in alphabetic order. An other essential fact is that because of the hash algorithm the user has to provide all characters of a given string to locate the entry in the master data set. In case one character is missing or misspelled, IMAGE will not be able to locate even an approximate location within the master data set.

Common arguments for/against IMAGE/KSAM

The question 'KSAM or 'IMAGE' divides the users into three groups: One group telling you that they are never using KSAM, an other group telling you that they will never use IMAGE, and a third group which is not sure. Users who have a strong opinion about KSAM and IMAGE have basic arguments against or for a special method. Some of them I have collected and present them here with some comments.

Argument one: IMAGE is more reliable than KSAM.

This argument has basically two reasons:

- KSAM files are ordinary MPE files, while IMAGE files are privileged files. 'Privileged' means that the data sets cannot be accessed via the file system for users who don't have privileged mode capability. The access to IMAGE files can only be done by the IMAGE subsystem. KSAM files can be opened by everyone with the so called 'NOKSAM' option.

- HP has spent more efforts implementing IMAGE than KSAM. This concerns data security as well as performance.

Comment: It might be true that IMAGE is some more reliable. It should also be considered that an internal error of IMAGE (destroyed pointer chains) causes more trouble than KSAM. In some cases, the entire IMAGE database is absolutely unreadable and has to be restored. In most cases, the KSAM data file is still accessable so that a backup is possible using the actual data.

Second argument: IMAGE has a security system, KSAM not.

IMAGE has a rather complicated system of password security. Entire data sets or single data items can be provided with two passwords, one for read and one for write access. KSAM doesn't have this facility.

Comment: There is no question that an access security can be provided by program internal functions. This, in some cases, means additional implementation efforts, but there is the advantage that the user can build his own security structure which is more propriate for the according application.

Third argument: IMAGE has more advanced locking facilities than KSAM.

IMAGE mainly provides locking on database level, data set level and data item level. The locking functions are quite complex and not very easy to use for big databases. KSAM doesn't provide any internal locking facility.

Comment: I have made the experience that there is nothing more misused than database locking. In many cases, there is no synchronisation and in other cases, the entire database is locked although only a part of it is accessed. A main problem with database locking is that a programming mistake is rarely discovered during testing. I myself prefer synchronisation of logical access functions rather than on database level which is applicable to IMAGE and KSAM and gives you a much better chance of testing.

Fourth argument: KSAM requires more space than IMAGE.

Unfortunately, this is mostly true. Although, there are cases where KSAM needs even less space than IMAGE - but this is an exception. The reason for this is:

- The file system builds the KSAM key file much larger than it actually needs to be. The reason is that it is unknown to the system how the key blocks in the key file are filled. There has to be enough space in the key file when the KSAM data file is full and the key blocks are only filled 50%.

- Doublicate keys in the data file are also stored double in the key file. When keys are repeated very often (like in our example the city 'Houston') the key is stored redundantly.

Comment: It can only be said that more and more disc space becomes available for the same price. I, however, know that this, sometimes, does not solve the problem.

Fifth argument: KSAM offers functions which IMAGE doesn't.

Because of the different structure of IMAGE and KSAM, there are two main functions available in KSAM which don't exist in IMAGE:

- KSAM allows access to data records when only parts of the key are provided. IMAGE always requires an entire key.

- Reading a KSAM file sequentially gives the records automatically sorted after their primary key. This allows to use a lot of functions like finding the next logical key in a row.

Comment: It is true that in some applications the use of IMAGE leads to great inconveniance. There are tools available which allow the user to copy and sort records out of IMAGE databases into a sequential file but, of course, this means additional overhead.

Sixth argument: KSAM files have to be unloaded and loaded again more often than IMAGE.

The reason for this is the difference in handling so called deleted records. The basic difference between IMAGE and KSAM is that in IMAGE, once a record has been deleted, the space is reusable to the system. In KSAM the records are marked as deleted records and don't appear in the key file, but the space within the data file is lost and can be made only reusable by unloading and loading the KSAM file.

Comment: There are alternate solutions to the deleted record problem in KSAM. Deleted records can be marked as deleted by the user himself without actually deleting them, using the FUPDATE intrinsic rather than the FREMOVE intrinsic. Nevertheless, in many cases this means a dramatic decrease on performance.

Seventh argument: Loading and unloading KSAM files takes a lot of time.

Well, it takes a lot of time for IMAGE databases too, but in KSAM it is worse. The reason here is again the method of how KSAM adds records into a KSAM file. Adding a record to a KSAM file always means that the KSAM system has to go through the entire tree of the KSAM key file to get the right position for the key item. After this position has been found, it might happen that the key block, in which the key item has to be stored, is already full. In this case, a so called block split is performed which means that half of the block is copied to an other location within the key file, and, of course, all pointers have to be maintained. FCOPY, for example, is able to add approx. 10 records per second into an existing KSAM file with five keys defined.

Comment: There is a software tool available (COPYRITE) which loads KSAM files up to 20 times faster than the file system does. In this case, KSAM files can be loaded even faster than IMAGE databases, using DBLOAD.

How does COPYRITE work?

COPYRITE takes a completely different approach to the problem of building KSAM files. Rather than perform several random access operations on each record (add record to data file, add key#1 value to key file, add key#2 value to key file, etc.), COPYRITE separates the process into multiple sequential activities.

First, COPYRITE copies all records to the KSAM data file using very fast I/O techniques (no-buf, multi-record access). Next, COPYRITE sorts all values for key #1 using the system sort, and then performs a series of sequential writes to add this information to the key file. If more than one key is specified, COPYRITE repeats the sort operation for each additional key.

If you choose to operate in privileged mode, COPYRITE uses additional techniques to improve performance. In this mode, COPYRITE extracts the key values for the sorts while copying the data file records. It uses no-wait I/O for maximum throuthput, and very fast memory operations to move data between the extra data segments used as buffers and the stack.

TINGLER

or

What sort of life do programs lead at run-time?

Lance Carnes
TeXeT Company
163 Linden Lane
Mill Valley, California 94941 U.S.A.

ABSTRACT: Tingler is a run-time performance measurement tool which shows the exact behavior of program code and permits significant performance improvements. It measures the run-time behavior of a program in a completely unique fashion: **it implants monitoring code into your program file, then runs the program in its normal fashion.** The monitoring code, which runs in user mode, collects a file of data on every call to and exit from procedures and COBOL paragraphs. This data is analyzed off-line and a report is printed showing: the total time spent in each procedure or COBOL paragraph; all calls into and out of each procedure or COBOL paragraph and the frequency; and the grand total of calls, segment switches, time spent in user code, and time spent in externals. Using this data, it is possible to optimize frequently used code, reduce inter-segment transfers, and make more efficient calls to externals and system intrinsics. The Appendix contains several sample analyses of test programs and HP software (COBOL and SPL compilers, and EDITOR).

INTRODUCTION

Programs which are used frequently should run efficiently. TeX82 is a text formatting/typesetting package transported from large 32-bit machines to run on the HP3000, and while this system is run frequently, it does not run fast. This is the story of optimizing TeX82 and, as it turns out, the story of optimizing any program on the HP3000.

The objectives were 1) to find out where the time was being spent (wasted), and 2) see what could be done about it. It was only after making several "guesses" as to the location of the bottlenecks in the system, implementing the optimizations and getting virtually no payoff, that it became apparent some sort of performance monitoring tool was needed. Guessing is not a bad technique, and it is surprising how much you really know about a program's operation. Equally surprising is how much you don't know, and this is where a performance tool is needed to help out.

What optimizations can you make?

Given the existing computer hardware and operating system, the goal is to make the best, most efficient use of these resources. Avoid unnecessary operations and use the best methods.

Optimize procedural code. Program code which is executed frequently should be carefully done. Notice the term 'frequently'. It is not necessary to optimize *every* module. Find the ones which are used a high percentage of the time, and concentrate on these. In the case of TEX82, there are over 600 procedures, and it would take an extremely patient person to optimize each one.

Avoid operating system overhead. On the HP3000, MPE shares resources with multiple tasks running at the same time. In particular, memory resources for holding program code and data are swapped on and off the disk. If the **data segment is kept small** as possible, there is less overhead to save off and restore this area when your process is inactive. If the **size of code segments is kept small** less time is required to load parts of the program. Since the program code is virtual, and since MPE and the firmware keep track of transfers needed between code segments, **reduce the number of transfers between code segments.** Since (nearly) all programs need to use system intrinsics, for example FREAD and FWRITE to read and write files, these **intrinsics should be used in the most efficient manner possible.**

Eliminate incorrect and unnecessary code. This means code which either gives the wrong answer or performs more iterations than needed. This is especially costly when it coupled with a slow resource, for example, updating a 1000 record file by issuing 2000 FWRITEs!

What information is needed?

Another way to put this is, if you are going to make a difference in the program's operating efficiency as discussed above, what sort of data about the program is required?

To learn something about procedural code bottlenecks, you need to know the **location in the program where the most time is being spent.** If your program has small modules, it is enough to know which procedure or paragraph is occupying the time. If your program consists of straight-line code, you will need to know the exact location(s) rather than the module name.

When trying to optimize operating system overhead several factors are needed. **Data segment size** may be reduced in various ways, including local declarations (if your programming language and compiler allow, e.g. COBOL74 does not, SPL and Pascal do), keeping literals in the code segment rather than in the data segment, and packing data as carefully as possible (e.g. PACKED types in Pascal). The relationship of code and the data required for execution of the code is a complicated one. On the HP3000, where the maximum direct data area is 64Kb (too small for TEX82 which uses 1Mb!), this factor is of less import than the following.

Code segment size and inter-segment transfers, if reduced, will improve performance. Here it is important to place the most often used code in small code segments, and to place modules which call each other frequently in the same module. To optimize in this area, you need to know **1) the high-use modules**, and **2) the most frequent inter-module calls.**

When optimizing the use of **system intrinsics**, since you cannot (usually) rewrite and re-segment them into your code file, you must find which ones you use and make the best use of them. To optimize here you need to know **which intrinsics are used, how often, which modules call them, and how much time is spent executing them.**

Incorrect or unnecessary code needs to be detected and then eliminated. When it is discovered, for example, that 2000 FWRITES are being issued to update a 1000 record file, the source of this should be tracked down and eliminated. But to get to this point, you must first find a way to detect that 2000 FWRITEs are being issued.

For optimizing TEX82, the following were selected as the data needed:

- frequency of module usage (since modules are small, this is sufficient);
- frequency and location of inter-module transfers;
- frequency and location of inter-segment transfers;
- frequency and location of **intra**-segment transfers (so that code segments with well-placed modules will be detected);
- frequency and location of calls to externals and system intrinsics;
- total time spent in each module or intrinsic.

Optimizing the use of the data segment has been deferred in the case of TEX82. Data segment size was not felt to be as important as the above factors, since a virtual memory environment was constructed which uses nearly all of the 64Kb data segment for the working set.

Finding the Bottlenecks: The Artful Guessing Game

When you want some answers, the first inclination is to consult the experts. The following is an example of the gospel on program performance.

> Program code which is not logically segmented makes it harder for the memory manager to do its job, causing disc accesses to be used for unnecessary swaps. Proper code segmentation is a complex topic (more like an art than a science), but here is a simplified training course: write modular code; don't segment until you have 4000 words of code; isolate modules that seldom run; isolate modules that often run; aim for 4000 words per segment, and group modules by "time" rather than "function;" if you reach 63 segments, increase segment size, but keep active segments smaller than inactive ones.
>
> (Robert M. Green, "Overview of Optimizing (On-Line and Batch)". From *Proceedings*, 1982 International Meeting, HP3000 IUG.)

While the rules suggested here are indisputable, no measurement technique is offered for finding "modules that often run" or for locating modules that occupy the most time. Consider also the following.

> The oldest method available to the designer to segment his/her application code is to use the eye. This technique is dependent upon the experience of the individual using it. It is prone to error. The technique is time consuming. It is always biased.
>
> ...
>
> Placing counters in the code is a way of gathering information to help in the determination of appropriate segmentation. By judicious placement of the counters, we may determine an execution profile to assist us in proper segmentation. However, this technique is fraught with problems as well. The counters need to be initialized; they will need debugging; they will require stack space; and they will interfere with execution.
>
> (Kim Leeper, "LOOK/3000: A New Real-Time System Performance Monitoring Tool". From *Proceedings*, 1982 International Meeting, HP3000 IUG.)

"Counters in the code" seems to be exactly what is required to collect the data needed for optimization, yet here we are warned away from this technique. Tingler, in fact, uses just such a technique and the inherent problems have been overcome, as is shown below.

What information is provided by performance monitors?

Of the available tools for local program optimization (as opposed to total system optimization) there seemed to be two types: 1) **static analyzers** and 2) **sampling monitors**.

Static analyzers use the source listing or the object file to report information about a program. Typically they will cross-reference all procedures and calls into and out of them. They may also show calling hierarchies, data references, number of times a particular instruction is used, and more. While these are useful, they are static and do not reflect any of the characteristics of the performing program.

Two examples of static analyzers are PROGINFO and PROGSTAT from the HPIUG Contributed Library.

Sampling monitors are programs which are executing at the same time as the program being measured. At regular time intervals, the sampler interrupts the target program and records the location in the program when the interrupt occurred. After a large number of samples have been taken, a statistical graph may be drawn showing the relative frequency of execution of various parts of the program. These locations may be matched up with symbolic names, linking them back to the source program. LOOK/3000 (Wick Hill Associates) and APS/3000 (Hewlett-Packard) are two examples of sampling monitors.

Samplers provide valuable data, but not enough to indicate where re-segmentation may provide a payoff. HP wrote this about APS/3000:

> For resegmentation purposes, APS/3000 will allow you to estimate the number of transfers of control between segments during program execution. Because it is a statistical sampling technique, APS/3000 assesses system operation at pre-set time intervals, rather than continually. Thus it will not provide the exact number of transfers, nor can it tell you specifically what to do to make your segmentation more efficient.
>
> (MPE Segmenter Reference Manual, Part No. 30000-90011. Hewlett-Packard Company, 11/82)

If you glance back at the data needed for optimization of TEX82, it is a combination of the items from static analyzers and monitoring samplers. We want a cross-reference of module linkages, both forward and backward. We need also the actual run-time call frequencies, and time spent in each module. If the monitoring sampler could put a frequency count and elapsed time on the cross-reference produced by the static analyzer, we would have fulfilled nearly all the requirements.

TINGLER!

Reviewing the required information, listed previously, what are the raw data items that need to be collected at run-time? It turns out all of the information required could be reduced from data on

- procedure and paragraph transfers, where they were called from and where they call to; and
- the processor and elapsed time when each of these events occurred.

Using this data, it is possible to reconstruct the factors needed: frequency and location of all transfers, and time spent within modules. The question now becomes, how can this data be collected?

The first idea was to have the compiler emit monitoring code for each procedure, or use an automated tool which would insert code into the source file. This seemed to be a complicated method, and others were considered.

Another idea was to use the TRACE bit in the PCAL and EXIT microcode. In fact the product TRACE/3000 from HP uses this, and for SPL and FORTRAN will show a blow-by-blow trace of a program, its module transfers and data values. TRACE/3000 does not show timings, and even if it did, it is not certain the actual program timings could be distinguished from TRACE/3000 overhead.

Then, one clear day, an idea presented itself which would allow collection of the data in a clean and reliable manner. This idea also had the side effect that it would work with **any** HP3000 Program file.

Every program file on the HP3000 consists of one or more code segments and transfers must be made between these code segments. The HP3000 architecture relies on a Segment Transfer Table (STT), a list of connecting code segments and entry points which lies at the end of each and every code segment. Whenever a PCAL is executed, the firmware finds the segment number and entry point in the STT, and then causes a transfer to that point.

Tpatcher, the program which implants the monitoring code, places a copy of it in each code segment, and then revectors the STT list so that the monitoring code is executed instead of taking the normal transfer. Once in the monitor code, the required data is noted (called-from address, destination module, and time of occurrence). The actual transfer is then executed.

Paragraph calls in COBOLII (COBOL74 compiler) are handled in a different manner, since they do not transfer via the STT. The machine instructions which handle the transfers are trapped using a different technique, and the requisite data about the transfer is noted (called-from addr, called-to segment and addr, and time).

There are several advantages to this method.

- No special compiles or PREPs are required;
- It will analyze programs with or without any supporting symbolic data (e.g. PMAP or compiler listing); in fact, system utilities such as compilers, editors, text formatters, file copiers, as well as applications may be monitored **using only the Program File** (see the Appendix);
- The method is clean, and no guesswork is required to locate the PCALs and EXITs, since they are trapped at their natural execution location, the STT.
- Since the monitoring code resides in the code segment no inter-segment transfer is required to do data collection. In addition, this makes it possible to accurately measure the overhead of the monitoring code.
- Since every transfer is intercepted, the resulting data is a precise picture of performance, and not a statistical estimate as is gotten with sampling monitors.

Capturing the data is one thing - analyzing it is another. An analyzer, called **Tlyzer**, has been implemented which reports:

- For each segment, the total time spent, location and frequency of transfers into the segment and out to other segments and intrinsics;

- For each procedure, the frequency and location of calls into and out of the procedure, and the total time spent (PMAP must be supplied);
- For each COBOL paragraph, the frequency and location of calls into and out of the paragraph, and the total time spent (COBOL compiler MAP listing and PMAP must be supplied);
- For each transfer frequency reported, those which require a code segment transfer are noted with an asterisk;
- A grand total of: procedure calls, paragraph calls, external calls, inter-segment transfers, and the time spent for each.

For a sample report, see the Appendix.

What Optimizations Will TINGLER Permit?

The following explains the optimizations which can be achieved as a result of analysis by TINGLER.

Procedural code bottlenecks are noted at the module level. If your code is modular, this is close enough to to permit inefficient code to be identified. (If you notice a particular section of code is inefficient and if your code is not modular, or the compiler you use does not produce modular code, Tingler will not be able to pinpoint the exact location(s).)

Code segment switches may be detected (even in absense of any supporting symbolic data, e.g. program listing or PMAP). Any program may be analyzed for transfers between segments, and transfers to externals or system intrinsics. This data alone may be sufficient to indicate that no further optimization is needed (e.g. if there is only one code segment, or if all significant time is spent in intrinsics). It may show that work is needed to better segment the application, or make better use of externals and intrinsics.

Incorrect or unnecessary code may be detected and removed or corrected. The programmer may use the information provided by Tingler and verify that it reflects the intention of the program. For example, if it is noticed that 2000 FWRITES are being called to update a file having only 1000 records, the discrepancy must be resolved. The double writes may be justified, and they may not.

A useful side-effect is using Tingler to track down bugs. Since it operates by recording a full trace of program flow, the programmer may play back the events blow-by-blow to determine where the logic has gone awry.

SUMMARY

There will always be a need to know more about the run-time behavior of computer programs, in order to optimize their performance and verify their function. Static analyzers and sampling monitors are useful but limited, especially in the HP3000 environment where virtual code management is integral to the hardware and software.

Tingler fulfills the need to know the exact program flow relative to time. Tingler permits resegmentation of programs by giving precise accountings of execution time and frequency. Tingler also permits verification of program function, and location of program logic errors.

Appendix

Here are some examples of the reports produced by Tingler, version 0.1. Tingler is still under development, and there are some important refinements yet to be made. As you look over these samples, keep these points in mind:

▶ **All timings include Tingler measurement overhead.** This is fairly significant (200 microseconds per event), especially for poorly segmented programs or programs which make a large number of transfers. In the next version of Tingler, this overhead time will removed from the actual times, and a more accurate picture will result. Note, however, that the times given here, while inflated, are each inflated by the **same amount**. Therefore, they may be used as a relative measure.

▶ **Fractions of milliseconds were not measured and are therefore not precise.** Note that the timings listed are in fractions of milliseconds. They will add up to give the totals listed in the Grand totals, for example, all segment times will add up to the exact total processor time. The reason fractions were kept is that many procedures execute in less than one millisecond. When the processor clock changes, the events which occurred since the last clock change are each given their increment of the millisecond. For example, if events A B C occur within one millisecond, events A and B are given 0.33 ms, and event C is given 0.34 ms. This way, procedures which hardly do anything, but which are called many times, will show up.

▶ COBOL paragraphs are not yet itemized. They are currently trapped by the monitor code and written to the run-time data file, but are not analyzed by Tlyzer. The next version will include this feature.

You are welcomed to communicate to me any suggestions or comments on any aspect of Tingler.

Lance Carnes
TeXTCompany
163 Linden Lane
Mill Valley, CA 94941 U.S.A.
Tel: (415) 388-8853
Telex: 910-481-0421

```
****                                                                    ****
****   The first two examples are simple and are intended to illustrate ****
****   Tingler's function.                                              ****
****                                                                    ****
****   The following SPL program was used for the test.                 ****
****                                                                    ****
****   The first example has no PMAP available, and shows performance   ****
****   by segment.                                                      ****
****                                                                    ****
****   The second example uses the PMAP and shows a further breakdown   ****
****   by segment. (procedure.                                          ****
****                                                                    ****
```

```
00000 0   begin  << spl test >>
00000 1
00000 1    intrinsic proctime,clock;
00000 1
00000 1   $control segment = seg'clocks
00000 1
00000 1   procedure get'proctime;
00000 1     begin tos := proctime;
00002 2     end;
00000 1
00000 1   procedure get'clock;
00000 1     begin tos := clock;
00002 2     end;
00000 1
00000 1   $control segment=seg'caller
00000 1
00000 1   procedure caller;
00000 1     begin integer i;
00000 2       for i := 1 until 1000 do
00006 2         begin get'proctime;
00012 2           if logical(i) then get'clock << every odd time >>
00014 2         end;
00016 2     end;
00000 1
00000 1   $control segment=main
00000 1
00000 1   caller;
00001 1
00001 1   end.
```

This program calls
PROCTIME 1000 times
and calls CLOCK
500 times.

Note the weird segmentation

```
****                                                                    ****
****  Example 1. PMAP not available.                                    ****
****                                                                    ****
****



** Tlyzer vers 0.1 (c) TeXeT Co. 1983 **

Program file analyzed: TNGTST.PASCAL.TEX

**** No PMAP available ****

Key to Symbols:

Seg  - Code Segment     Proc - Procedure     Ext  - External or Intrinsic
 *   - Indicates that code segment transfer was needed


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  0)                   (%00)  total time (ms) =      42.17  (6.51 %)

Number of Times Called By:

Number of Calls To:
    Seg  (  1)                     =     1*     Ext  (  2) TERMINATE'        =     1

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  1)        *          (%01)  total time (ms) =     147.32  (22.73 %)

Number of Times Called By:
    Seg  (  0)                     =     1*
Number of Calls To:
    Seg  (  2)                     =  1500*
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  2)                   (%02)  total time (ms) =     307.84  (47.48 %)

Number of Times Called By:
    Seg  (  1)                     =  1500*
Number of Calls To:
    Ext  (  0) PROCTIME            =  1000     Ext  (  1) CLOCK             =   500

* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*********************

Ext  (  0) PROCTIME          total time (ms) =     98.22  (14.05 %)

Number of Times Called By:
    Seg  (  2)                     =  1000

*********************

Ext  (  1) CLOCK             total time (ms) =     54.00  (8.44 %)

Number of Times Called By:
    Seg  (  2)                     =  500

*********************

Ext  (  2) TERMINATE'        total time (ms) =      0.00

- - - - - - - - - - - - - - - - - - - - - - -
Grand totals:

Total processor time (ms) =           648
   Time in user code (ms) =           497
   Time in externals (ms) =           151
Total elapsed time=                  1200
Total # procedure calls=             3002
Total inter-segment transfers=       3003
# Process interruptions=               15
```

Break down by segment.

See Example 2 for procedures analysis.

as expected!

```
****                                                                ****
****  Example 2. PMMP available.                                    ****
****                                                                ****
****

** Tlyzer vers 0.1 (c) TeXeT Co. 1983 **

Program file analyzed: TNGTST.PASCAL.TEX

Key to Symbols:

Seg  - Code Segment     Proc - Procedure     Ext  - External or Intrinsic
 *   - Indicates that code segment transfer was needed



- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  0) MAIN            (%00)  total time (ms) =      45.01  (6.95 %)

Number of Times Called By:

Number of Calls To:
    Seg  (  1) SEG'CALLER        =     1*     Ext  (  2) TERMINATE'       =    1
```

Outer Block

```
*********************
Proc (  0) OB'            %00.%00000 total time (ms) =      45.01  (6.95 %)

Number of Times Called By:

Number of Calls To:
    Proc (  1) CALLER            =     1*     Ext  (  2) TERMINATE'       =    1

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  1) SEG'CALLER      (%01)  total time (ms) =     152.11  (23.47 %)

Number of Times Called By:
    Seg  (  0) MAIN              =     1*
Number of Calls To:
    Seg  (  2) SEG'CLOCKS        =  1500*

*********************
Proc (  1) CALLER         %01.%00000 total time (ms) =     152.11  (23.47 %)
```

Caller

```
Number of Times Called By:
    Proc (  0) OB'               =     1*
Number of Calls To:
    Proc (  3) GET'PROCTIME      =  1000*     Proc (  2) GET'CLOCK        =   500*

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  2) SEG'CLOCKS      (%02)  total time (ms) =     295.97  (45.67 %)

Number of Times Called By:
    Seg  (  1) SEG'CALLER        =  1500*
Number of Calls To:
    Ext  (  0) PROCTIME          =  1000      Ext  (  1) CLOCK            =   500

*********************
Proc (  2) GET'CLOCK      %02.%00000 total time (ms) =     101.89  (15.72 %)
```

Get'Clock

```
Number of Times Called By:
    Proc (  1) CALLER            =   500*
Number of Calls To:
    Ext  (  1) CLOCK             =   500

*********************
Proc (  3) GET'PROCTIME   %02.%00003 total time (ms) =     194.08  (29.95 %)
```

Get'Proctime

```
Number of Times Called By:
    Proc (  1) CALLER            =  1000*
Number of Calls To:
    Ext  (  0) PROCTIME          =  1000
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

*********************
Ext  (  0) PROCTIME        total time (ms) =     102.44  (15.81 %)

Number of Times Called By:
    Proc (  3) GET'PROCTIME      =  1000

*********************
Ext  (  1) CLOCK           total time (ms) =      52.48  (8.10 %)

Number of Times Called By:
    Proc (  2) GET'CLOCK         =   500

*********************
Ext  (  2) TERMINATE'      total time (ms) =       0.00
- - - - - - - - - - - - - - - - - - - - -
Grand totals:

Total processor time (ms) =         648
   Time in user code (ms) =         493
   Time in externals (ms) =         155
Total elapsed time=                1637
Total # procedure calls=           3002
Total inter-segment transfers=     3003
# Process interruptions=             18
```

Note the difference from Example 1.
The same program was taken but
in two separate [illegible].

```
****                                                                    ****
****   COBOL compiler.                                                  ****
****                                                                    ****
****   This is an analysis of a 1600 source line compile.               ****
****                                                                    ****
****   Tingler shows that this is a well-segmented program,             ****
****   for the most part.                                               ****
****                                                                    ****
****   Note the well-segmented parts, and the poorly-segmented parts.   ****
****                                                                    ****
****   This is a partial printing. If space allowed in this publication, ****
****   you would see performance data for all segments and externals.   ****


** Tlyzer vers 0.1 (c) TeXeT Co. 1983 **

Program file analyzed: COBOLII.PASCAL.TEX

**** No PMAP available ****

Key to Symbols:

Seg  - Code Segment     Proc - Procedure     Ext  - External or Intrinsic
 *   - Indicates that code segment transfer was needed


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  0)                    (%00)  total time (ms) =     1811.29  (1.90 %)

Number of Times Called By:
     Seg  (  0)                      =  3048     Seg  (  4)                  =    86*
     Seg  ( 18)                      =  7316*    Seg  ( 19)                  =    24*
     Seg  ( 21)                      =     2*    Seg  ( 23)                  =   159*

Number of Calls To:
     Seg  (  0)                      =  3040     Seg  (  1)                  =     4*
     Seg  ( 18)                      =     4*    Seg  ( 19)                  =     4*
     Seg  ( 21)                      =     1*    Ext  ( 52) FOPEN            =     2
     Ext  ( 53) FCHECK               =     1     Ext  ( 54) FCLOSE           =     1
     Ext  ( 55) FGETINFO             =     2     Ext  ( 56) FWRITEDIR        =   105
     Ext  ( 57) FREADDIR             =    82

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  1)                    (%01)  total time (ms) =    14483.01  (15.18 %)

Number of Times Called By:
     Seg  (  0)                      =     4*    Seg  (  1)                  = 38688
     Seg  (  2)                      =   500*    Seg  (  3)                  =   358*
     Seg  (  4)                      =   454*    Seg  (  5)                  =  3696*
     Seg  (  8)                      =     3*    Seg  (  8)                  =  1738*
     Seg  ( 10)                      =  1317*    Seg  ( 11)                  =     8*
     Seg  ( 12)                      =     3*    Seg  ( 13)                  =  2462*
     Seg  ( 14)                      =  3281*    Seg  ( 15)                  =  4640*
     Seg  ( 18)                      =     3*    Seg  ( 19)                  =   269*
     Seg  ( 20)                      =   281*    Seg  ( 21)                  =     8*
     Seg  ( 22)                      =    10*    Seg  ( 23)                  =  1481*
     Seg  ( 24)                      =     1*    Seg  ( 27)                  =     9*
     Seg  ( 28)                      =     1*    Seg  ( 29)                  =     1*
     Seg  ( 31)                      =  1761*    Seg  ( 32)                  =     1*
     Seg  ( 33)                      =    31*    Seg  ( 34)                  =  8045*
     Seg  ( 35)                      =     1*    Seg  ( 42)                  =   755*
     Seg  ( 43)                      =     1*
Number of Calls To:
     Seg  (  1)                      = 38688     Ext  ( 50) DLSIZE           =    13
     Ext  ( 51) STACKSIZE            =    28     Ext  ( 52) FOPEN            =     3
     Ext  ( 58) FWRITEDIR            =   348     Ext  ( 57) FREADDIR         =   583
```

well-segmented!

(Parts of printout were deleted for publication.)

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  7)                    (%07)  total time (ms) =    10377.20  (10.88 %)

Number of Times Called By:
     Seg  (  7)                      = 12605     Seg  ( 21)                  =    11*
     Seg  ( 25)                      = 47872*    Seg  ( 38)                  =    62*
     Seg  ( 39)                      =   57*     Seg  ( 40)                  =     3*
     Seg  ( 41)                      =   109*
Number of Calls To:
     Seg  (  7)                      = 12605     Ext  ( 50) DLSIZE           =     4
```

Poorly-segmented. '*' means code segment change needed.

```
*********************

Ext  ( 56) FWRITEDIR          total time (ms) =     1390.93  (1.46 %)

Number of Times Called By:
     Seg  (  0)                      =   105     Seg  (  1)                  =   349


*********************

Ext  ( 57) FREADDIR           total time (ms) =     1945.44  (2.04 %)

Number of Times Called By:
     Seg  (  0)                      =    62     Seg  (  1)                  =   583
     Seg  ( 19)                      =     1     Seg  ( 42)                  =     4


- - - - - - - - - - - - - - - - - - - -
Grand totals:

Total processor time (ms) =         95391
   Time in user code (ms) =         88804
   Time in externals (ms) =          6587
Total elapsed time=                192265
Total # procedure calls=           349074
Total inter-segment transfers=     252831
# Process interruptions=             3707
```

interesting ratio

```
****  SPL compiler                                                    ****
****                                                                  ****
****  This is the analysis by Tingler of a 585 source line compile.  ****
****                                                                  ****
****  Though generally well-segmented, there is still a lot of       ****
****  thrashing between code segments.                                ****
****                                                                  ****

** Tlyzer vers 0.1 (c) TeXeT Co. 1983 **

Program file analyzed: SPL.PUB.SYS

**** No PMAP available ****

Key to Symbols:

Seg  - Code Segment     Proc - Procedure     Ext  - External or Intrinsic
 *   - Indicates that code segment transfer was needed


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  ( 21)                 (%25)  total time (ms) =     425.32  (3.94 %)

Number of Times Called By:
     Seg  (  0)                 =    19*    Seg  (  3)                 =   21*
     Seg  (  4)                 =    84*    Seg  (  5)                 =   37*
     Seg  (  7)                 =     2*    Seg  (  8)                 =  434*
     Seg  (  9)                 =    37*    Seg  ( 11)                 =   61*
     Seg  ( 13)                 =    13*    Seg  ( 15)                 =   15*
     Seg  ( 18)                 =   174*    Seg  ( 21)                 =  845
     Seg  ( 22)                 =    71*
Number of Calls To:
     Seg  ( 21)                 =   845     Seg  ( 22)                 =  953*

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  ( 22)                 (%26)  total time (ms) =    3258.30  (30.20 %)

Number of Times Called By:
     Seg  (  0)                 =     1*    Seg  (  1)                 =  178*
     Seg  (  2)                 =     7*    Seg  (  3)                 =   21*
     Seg  (  4)                 =   250*    Seg  (  5)                 =   63*
     Seg  (  6)                 =   637*    Seg  (  7)                 =   20*
     Seg  (  8)                 =   364*    Seg  (  9)                 =  284*
     Seg  ( 10)                 =    49*    Seg  ( 11)                 =  656*
     Seg  ( 12)                 =   488*    Seg  ( 13)                 =  141*
     Seg  ( 14)                 =   197*    Seg  ( 15)                 =   15*
     Seg  ( 16)                 =    80*    Seg  ( 18)                 =   28*
     Seg  ( 19)                 =   134*    Seg  ( 21)                 =  953*
     Seg  ( 22)                 =  4792     Seg  ( 23)                 =   86*
     Seg  ( 24)                 =  1034*    Seg  ( 25)                 =  135*
     Seg  ( 28)                 =   865*    Seg  ( 29)                 =   20*

Number of Calls To:
     Seg  ( 21)                 =    71*    Seg  ( 22)                 = 4792
     Seg  ( 23)                 =    25*    Seg  ( 24)                 = 1184*
     Ext  ( 17) FWRITE          =   833     Ext  ( 24) FREADDIR        =  132
     Ext  ( 25) FWRITEDIR       =   164
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  ( 28)                 (%34)  total time (ms) =    1788.10  (16.58 %)

Number of Times Called By:
     Seg  (  1)                 =   290*    Seg  (  2)                 =  136*
     Seg  (  3)                 =    67*    Seg  (  4)                 =  204*
     Seg  (  5)                 =    32*    Seg  (  7)                 =    4*
     Seg  (  8)                 =   875*    Seg  ( 10)                 =   33*
     Seg  ( 11)                 =   423*    Seg  ( 13)                 =  213*
     Seg  ( 15)                 =    61*    Seg  ( 18)                 =  132*
     Seg  ( 19)                 =   188*    Seg  ( 23)                 =   88*
     Seg  ( 28)                 =  5750     Seg  ( 29)                 =    1*

Number of Calls To:
     Seg  ( 22)                 =   865*    Seg  ( 25)                 =  148*
     Seg  ( 28)                 =  5750
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

*********************

Ext  ( 24) FREADDIR        total time (ms) =     386.29  (3.58 %)

Number of Times Called By:
     Seg  ( 22)                 =   132

*********************

Ext  ( 25) FWRITEDIR       total time (ms) =     425.12  (3.94 %)

Number of Times Called By:
     Seg  ( 22)                 =   164

- - - - - - - - - - - - - - - - - - - - - -
Grand totals:

Total processor time (ms) =         10784
   Time in user code (ms) =          8618
   Time in externals (ms) =          2166
Total elapsed time=                 53404
Total # procedure calls=            31348
Total inter-segment transfers=      27139
# Process Interruptions=              973
```

Good or bad segmentation!

Good segmentation!

```
••••  EDITOR.                                                     ••••
••••                                                              ••••
••••  This is the analysis of the following operation:            ••••
••••                                                              ••••
••••      TEXT A; GATHER ALL; KEEP B                              ••••
••••                                                              ••••
••••  Can you explain why FOPEN is called 13 times?               ••••
••••                                                              ••••


•• Tlyzer vers 0.1 (c) TeXeT Co. 1983 ••

Program file analyzed: EDITOR.PUB.SYS

•••• No PMAP available ••••

Key to Symbols:

Seg  - Code Segment     Proc - Procedure     Ext  - External or Intrinsic
 •   - Indicates that code segment transfer was needed

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  (  3)                  (%03)  total time (ms) =     1218.34  (21.31 %)

Number of Times Called By:
    Seg  (  3)                   =   585     Seg  (  7)               =    5•
    Seg  (  8)                   =     1•
Number of Calls To:
    Seg  (  3)                   =   585     Seg  (  4)               =    2•
    Seg  (  9)                   =     1•    Seg  ( 10)               =    3•
    Seg  ( 11)                   =  4323•    Ext  ( 18) ZSIZE         =    3
    Ext  ( 21) FREEDSEG          =     1     Ext  ( 24) DASCII        =  585
    Ext  ( 29) FGETINFO          =     3     Ext  ( 33) FOPEN         =    7
    Ext  ( 34) FCLOSE            =     8     Ext  ( 36) DATE'LINE     =    1
    Ext  ( 38) FCHECK            =     1     Ext  ( 39) FWRITE        =    5
```

Poorly segmented

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

Seg  ( 11)                  (%13)  total time (ms) =     1423.66  (24.90 %)

Number of Times Called By:
    Seg  (  3)                   =  4323•    Seg  (  4)               =    6•
    Seg  (  7)                   =     7•    Seg  (  8)               =    1•
    Seg  ( 10)                   =     8•    Seg  ( 11)               =    8

Number of Calls To:
    Seg  ( 11)                   =     8     Ext  (  0) FWRITEDIR     =  663
    Ext  ( 27) FREADDIR          =   737     Ext  ( 29) FGETINFO      =    3
    Ext  ( 32) FREAD             =     1     Ext  ( 39) FWRITE        =    3

•  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •  •

•••••••••••••••••••••

Ext  (  0) FWRITEDIR        total time (ms) =      751.71  (13.15 %)

Number of Times Called By:
    Seg  ( 11)                   =   663

•••••••••••••••••••••
•••••••••••••••••••••

Ext  ( 27) FREADDIR         total time (ms) =      627.49  (10.98 %)

Number of Times Called By:
    Seg  ( 11)                   =   737

•••••••••••••••••••••
•••••••••••••••••••••

Ext  ( 32) FREAD            total time (ms) =       48.50  (.85 %)

Number of Times Called By:
    Seg  (  4)                   =     7     Seg  ( 11)               =    1


•••••••••••••••••••••

Ext  ( 33) FOPEN            total time (ms) =      314.63  (5.50 %)
```

← FOPEN called 13 times!?

```
Number of Times Called By:
    Seg  (  3)                   =     7     Seg  (  4)               =    4
    Seg  (  7)                   =     2

•••••••••••••••••••••

Ext  ( 34) FCLOSE           total time (ms) =      351.68  (6.15 %)

Number of Times Called By:
    Seg  (  3)                   =     8     Seg  (  4)               =    3
    Seg  (  7)                   =     1

•••••••••••••••••••••
•••••••••••••••••••••

Ext  ( 38) FWRITE           total time (ms) =      277.24  (4.85 %)

Number of Times Called By:
    Seg  (  3)                   =     5     Seg  (  4)               =   74
    Seg  ( 11)                   =     3

- - - - - - - - - - - - - -   - - - - - - - - -
Grand totals:

Total processor time (ms) =          5717
   Time in user code (ms) =          2971
   Time in externals (ms) =          2746
Total elapsed time=                 35858
Total • procedure calls=             7692
Total inter segment transfers=       8721
• Process interruptions=              600
```