

BUSINESS PROCEEDINGS

HP 3000/SERIES 100

VOLUME I



FOCUS
ON THE
FUTURE

INTEREX

1 9 8 6

DETROIT

INTEREX
DETROIT CONFERENCE
SEPTEMBER 28 - OCTOBER 3, 1986

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

INTEREX



the International Association of
Hewlett-Packard Computer Users

Proceedings of the 1986 Conference at

Detroit, Michigan

Hosted by the

Southeastern Michigan Users Group

Papers for the
HP3000
and
Series 100

Tom Léger, Editor

HOST COMMITTEE

Thomas Léger

Steve Gauss

Barb Gauche

Gib Spaman

Ed Witkow

Jim Leblanc

Terry Grover

Gary Dancy

Gary Latzman

Joe McGarry

Florence Moore

INTRODUCTION

It is with great pleasure and satisfaction that the Southeastern Michigan Users Group (SMUG), the host committee for the 1986 North American Interex Conference, presents the proceedings for this conference.

We wish primarily to thank all the authors who have labored long to share their expertise with us.

I want to thank the paper selection committee:

Jim Leblanc
Gary Dancy
Sharon Brevoort
Ed Witkow

who were assisted in reviewing over 200 submitted abstracts by:

Jeff Hansen and the
Lake Michigan Regional User Group and
Judy Cree and the
Toledo Users Group.

We are also grateful to Pam Tower and Barton Coddington of H.P. for their assistance with Hewlett Packard authors.

These proceedings are organized alphabetically by author within functional classification, systems specialists, application specialists and executive.

It is our sincere wish and primary objective that these proceedings help you to

FOCUS ON THE FUTURE!

Sincerely,

Conference and Business Program Chairman
Thomas Léger

Detroit Interex Conference
Business Proceedings By Classification
(HP3000/100)

1.1 SYSTEMS SPECIALISTS

1.1.1 Performance

| | | | |
|--|-----------------------|-------------------|------|
| APPLICATION PERFORMANCE TUNING - APS/3000 | JORDAN, ARTHUR | HEWLETT PACKARD | 3101 |
| DISC PERFORMANCE - WHAT IS IT ? | ALDINGER, RICK | HEWLETT PACKARD | 3102 |
| IDENTIFYING OPPORTUNITIES FOR PERFORMANCE IMPROVEMENT | SCOTT, GEORGE B. | ELOCOR CORP | 3103 |
| IMPACTS OF TECHNOLOGY ON HIGH PERFORMANCE MASS STORAGE | JAMES, DAVID | HEWLETT PACKARD | 3104 |
| IMPROVING YOUR PERFORMANCE | VAN VALKENBURGH, R.E. | AMPEX CORPORATION | 3105 |
| PROCESS HANDLING WITH BUSINESS BASIC | CRAIG, JACK | BRIDGEWARE | 3106 |
| STRATEGIES FOR EXTENDING THE LIFE OF THE HP 3000 | DOWLING, JAMES | BOSE CORPORATION | 3107 |

1.1.2 Advanced Systems Management

| | | | |
|--------------------------------|--------------|-----------------|------|
| HPE FILE SYSTEM OVERVIEW | KONDORFF, AL | HEWLETT PACKARD | 3108 |
| HPE VOLUME MANAGEMENT OVERVIEW | EDHART, RICK | HEWLETT PACKARD | 3109 |
| THE SYSTEM MANAGERS' TOOLBOX | BLAKE, ISAAC | HEWLETT PACKARD | 3110 |

1.1.3 Communications-Directions & Limits

| | | | |
|--|-----------------|-----------------|------|
| COMMUNICATING BETWEEN HP'S AND FOREIGN SYSTEMS | ATKINSON, TERRY | | 3111 |
| INTERPROCESS COMMUNICATION USING HPE MESSAGE FILES | BORRESEN, LARS | HEWLETT PACKARD | 3112 |

1.1.5 Focus on Your Future

| | | | |
|---|------------------|------------------------------|------|
| EXPERT SYSTEM MANAGER FOR THE HP3000 | HOPMANS, ROSS G. | BRANT COMPUTER SERVICES LTD. | 3113 |
| THE ROLE OF THE OPERATOR IN THE DATA CENTER OF THE FUTURE | DRAKE, RONALD | OPERATIONS CONTROL SYSTEMS | 3114 |

1.1.6 Production Processing Control

| | | | |
|--|--------------------|----------------------------|------|
| PROTECTING YOUR SOFTWARE INVESTMENT: AN AUTOMATED APPROACH | LEVY LEIGHT, BETSY | OPERATIONS CONTROL SYSTEMS | 3115 |
|--|--------------------|----------------------------|------|

1.1.7 Disaster/Backup/Recovery

| | | | |
|--|---|--|--------------|
| AUTOCHANGER EXTENDS CARTRIDGE TAPE DRIVE CAPACITY | ESCUDE, MANUEL | HEWLETT PACKARD | 3116 |
| BACKUP SYSTEMS, PRESENT AND FUTURE | HUFFMAN, JACK | HEWLETT PACKARD | 3117 |
| INSURING THE FUTURE OF YOUR DATA BY CONTINGENCY PLANNING | KAMINSKI, THOMAS J. VIRGILIO, LESLIE A. GROSSLER, JOERG | SINGER EDUCATION DIVISION SINGER EDUCATION DIVISION JOERG GROSSLER OMB | 3118 3119 |

1.1.8 Networking and Connectivity

| | | | |
|---|--------------------|-----------------|------|
| HEWLETT PACKARD ON COMMUNICATIONS PREMISES WIRING | DUDLEY, KAREN | HEWLETT PACKARD | 3120 |
| HP ADVANCEMET - INTEGRATING NEW PROCESSORS | RICHARDSON, STEVE | HEWLETT PACKARD | 3121 |
| ISDN NETWORKING FOR THE OFFICE | SHAPE, TIM | HEWLETT PACKARD | 3122 |
| MAP (MANUFACTURING AUTOMATION PROTOCOL) | ESTES, ROBERTA | HEWLETT PACKARD | 3123 |
| WIDE AREA NETWORKING - A CASE STUDY | HELLEBOLD, OLIVIER | HEWLETT PACKARD | 3124 |
| X.25: WHAT TO DO AFTER THE NETWORK IS IN PLACE | COTA, STEPHEN J. | MCI DISC | 3125 |

1.1.9 Spectrum/RISC

| | | | |
|--|---|--|--------------|
| A COMPARISON OF C COMPILERS FOR THE HP3000 | STELER, STAN | ALLEGRO CONSULTANTS, INC. | 3126 |
| COMMERCIAL SPECTRUM PROGRESS REPORT | SIMON, RICK | HEWLETT PACKARD | 3127 |
| MEETING THE CHALLENGE: AN INSIDE LOOK AT SPECTRUM TESTING | BARNETT, DONALD HOLT, WAYNE E. | HEWLETT PACKARD SOFTWARE RESEARCH NORTHWEST | 3128 3129 |
| MIGRATING TO THE SERIES 900'S - VARIABLES AFFECT. SYS. PERF. | FRIEDRICH, RICHARD McBRIDE, BECKY | HEWLETT PACKARD HEWLETT PACKARD | 3130 3131 |
| MIGRATION SOLUTIONS FOR HPE/XL | CADOMONI, LAWRENCE J. GARCIA, I. JAMET | HEWLETT PACKARD HEWLETT PACKARD | 3130 3131 |
| HPE XL ORGANIZATION AND DIRECTION | COURTNEY, LEE | HEWLETT PACKARD | 3131 |
| PROGRAMMING THE NEW GENERATION OF HP COMPS.: A RISC TUTORIAL | HECKER, JEFFREY | ARENS APPLIED ELECTROMAGNETICS | 3132 |

Detroit Interex Conference
Business Proceedings By Classification
(HP3000/100)

1.2 APPLICATIONS SPECIALISTS

1.2.1 Information Center

| | | | |
|----------------------------------|-------------------|---------------------|------|
| 4GL - THE CONTROVERSY RAGES ON | HEATER, KAREN | INFOCENTRE LTD. | 3201 |
| INFORMATION CENTERS AROUND 4GL'S | REHILLARD, ROBERT | COGNOS INCORPORATED | 3202 |

1.2.2 File Transfer

| | | | |
|--|----------------|-------------------------|------|
| LETS SCOPE IT OUT - FILE TRANSFER BEYOND THE SWEAKER-NET | FOSTER, BIRKET | N.B.FOSTER & ASSOCIATES | 3203 |
|--|----------------|-------------------------|------|

1.2.3 4th Generation Performance Issues

| | | | |
|---|------------------------------------|--|------|
| ADVANCED TRANSACT PROGRAMMING TECHNIQUES | BUTLER, STEPHEN M. McINTOSH, JC | PROBUS INTERNATIONAL INC. PROBUS INTERNATIONAL INC. | 3204 |
| ENRICHING YOUR POWERHOUSE ENVIRONMENT | ROBINSON, DAVID G. | ROBINSON, WALLACE & COMPANY | 3205 |
| PEOPLE & MACHINE PERFORMANCE W/SPEEDWARE, POWERHOUSE, RAPID | BRAYMAN, CHRISTOPHER | BRAMT COMPUTER SERVICES LTD. | 3206 |
| PERSONAL COMPUTERS SOLVE 4GL PROBLEMS | WARZECHA, CHARLES E. | GATEWAY SYSTEMS CORPORATION | 3207 |
| THE FUTURE OF SYSTEM DEVELOPMENT | FROST, RICHARD A. | FUTURE IDEAS, INC. | 3208 |
| THE KEY TO UNLOCKING PEAK PERFORMANCE | CASEY-DAVIS, KIMBERLEE | KAISER | 3209 |
| THE SYSTEM LIFE CYCLE IN THE 4GL ENVIRONMENT | SOLLAND, LEIGH | COGNOS INCORPORATED | 3210 |

1.2.4 Graphics

| | | | |
|---|-----------------|-----------------|------|
| GRAPHICS IN AN ORGANIZATION | TEMPLE, YVONNE | HEWLETT PACKARD | 3211 |
| HP2680A, THE MYSTICAL PRINTER, HOW IT WORKS | OXFORD, RICHARD | MCI DISC | 3212 |

1.2.5 Focus on Your Future

| | | | |
|--|---------------------|------------------------------|------|
| 4GL & THE CHANGING ROLE OF THE PROGRAMMER | FARQUHARSON, IAN | INFOCENTRE LTD. | 3213 |
| A WINDOW INTO THE FUTURE | KOHON, MICHEL | TYLABS CORPORATION | 3214 |
| AN EXPERT FINANCIAL PLANNING SYSTEM | HOPMANS, ROSS G. | BRAMT COMPUTER SERVICES LTD. | 3215 |
| | HACKENZIE, DON | BRAMT COMPUTER SERVICES LTD. | 3216 |
| HOW TO DEVELOP NEW APPLICATIONS - A STRATEGY | WALLACE, MARK | ROBINSON, WALLACE & COMPANY | 3217 |
| THE MINI & THE MICRO-DISTRIBUTED APPLICATION DEVELOPMENT | FIORAVANTI, PATRICK | INFOCENTRE LTD. | 3217 |

1.2.6 After IMAGE

| | | | |
|--|------------------------|-----------------------------|------|
| IS THERE LIFE BESIDES IMAGE ? | KOVALICK, NAY | HEWLETT PACKARD | 3218 |
| PERFORMANCE PROGRAMMING WITH HPSQL/V | DEMERY, PAUL | HEWLETT PACKARD | 3219 |
| RELATIONAL DATABASE : HOW DO WE KNOW WE NEED ONE ? | LARSON, ORLAND | HEWLETT PACKARD | 3220 |
| RELATIONAL DATABASES VS. IMAGE: WHAT THE FUSS IS ALL ABOUT | VOLOKH, EUGENE | VESSOFT, INC. | 3221 |
| THE FUTURE OF DATABASE TECHNOLOGY | TRASKO, MARK S. | DYNAMIC INFORMATION SYSTEMS | 3222 |
| TRENDS IN IMAGE | TASHENBERG, C. BRADLEY | BRADMARK COMPUTER SYSTEMS | 3223 |

1.2.7 Tools

| | | | |
|--|---------------------|--------------------------------|------|
| COMPUTER ASSISTED QUALITY ASSURANCE FOR SOFTWARE DEVELOPMENT | ROSENBERG, JONATHAN | OPERATIONS CONTROL SYSTEMS | 3224 |
| LINKING TO HP SYSTEM DICTIONARY | HARNAR, RON | HEWLETT PACKARD | 3225 |
| PROTOTYPING AND SYSTEMS DEVELOPMENT USING 4GL | OUELLETTE, RAYMOND | INFOCENTRE LTD. | 3226 |
| RIMS, RIMS, RIMS : WHY, WHEN, AND HOW | BRUNO, BENEDICT G. | STR SOFTWARE COMPANY | 3227 |
| SOFTWARE DESIGN FOR LONG-TERM RELIABILITY & MAINTAINABILITY | TOBACK, BRUCE | OPT. INC. | 3228 |
| THE FOURTH BEAR OF IMAGE | WHITE, FRED | ADAGER | 3229 |
| THE SPIRIT OF A NEWER SOFTWARE : LL'SPIRIT | LIAT, LIM | SINGAPORE COMPUTER SYSTEMS PTE | 3230 |
| THE TOOLS OF STRUCTURED ANALYSIS - A TUTORIAL | WALLACE, MARK | ROBINSON, WALLACE & COMPANY | 3231 |

1.2.8 Office Automation

| | | | |
|--|----------------------|------------------------------|------|
| CREATIVE SYSTEM INTEGRATION TO ENHANCE PRODUCTIVITY | RUTHERFORD, JILL C. | BOEING AEROSPACE CO. | 3232 |
| ONLINE INFORMATION SERVICES FOR HP PERSONAL COMPUTER USERS | CROW, BILL | HEWLETT PACKARD | 3233 |
| PLANNING INTEGRATED OFFICE SYSTEMS | SCHRAH, W.P. | | 3234 |
| UTILIZING TELEPHONE USAGE DATA - AUTOMATICALLY | BURCHETT, RICHARD L. | INFOFLOW INTERNATIONAL, INC. | 3235 |

1.2.9 Spectrum/RISC

| | | | |
|--|-------------------|---------------------|------|
| MIGRATING COBOL PROGRAMS TO SPECTRUM: A BATTLE OR A BREEZE ? | SPENCE, STEVEN J. | HEWLETT PACKARD | 3236 |
| MIGRATING POWERHOUSE APPLICATIONS TO NEW MACHINE ENVIRONMENT | SINCLAIR, JIM | COGNOS INCORPORATED | 3237 |
| THE HPIMAGE INTERFACE COMPONENT OF ALLBASE | CHENG, SARA | HEWLETT PACKARD | 3238 |
| USING THE HPE/XL LINK EDITOR | COUTANT, CARY A. | HEWLETT PACKARD | 3239 |

Detroit Interex Conference
Business Proceedings By Classification
(HP3000/100)

1.3 EXECUTIVES

1.3.1 Corporate Culture

| | | | |
|--|-------------------|--------------------------------|------|
| ADOLESCENCE AT THE AGE OF 137:STUDY CHANGE CORPORATE CULTURE | INDERHILL,MARK | AMPAC DISTRIBUTION CORPORATION | 3301 |
| ARCHAEOLOGY OF THE HP3000 COMPUTER | GREEN,ROBERT M. | ROBELLE CONSULTING LTD. | 3302 |
| HOW TO DESIGN A CHINESE STYLE NETWORK SYSTEM | HWANG,HEH-JET | | 3303 |
| POWER TO THE PEOPLE, EXPERIENCES OF A START-UP DP MANAGER | LAZAR,CLIFFORD W. | SYSTEMS EXPRESS | 3304 |
| THE SECOND MARKET : BUYING AND SELLING USED HP EQUIPMENT | BUCHANAN,BUCK | FIDELITY SYSTEMS | 3305 |
| WHAT HP DIDN'T TELL YOU WHEN YOU BOUGHT YOUR HP3000 | WISEMAN,D.B. | SYSTEM SOFTWARE LIMITED | 3306 |

1.3.2 M.I.S.

| | | | |
|---|-------------------------|--------------------------------|------|
| DATABASE DESIGN IN AN IMPERFECT WORLD | SHOWARD CLARKSON,MARCIA | THE UNIVERSITY OF THE SOUTH | 3307 |
| MANAGEMENT - THE FORGOTTEN PART OF M.I.S. | EDWARDS,BRUCE | BRIDGE OIL LIMITED | 3308 |
| METHODS AND PRACTICES OF MIS MANAGEMENT | KLEINMAN,MITCHELL | CONSOLIDATED CAPITAL COMPANIES | 3309 |
| STRATEGIC PLANNING IN SMALL MIS SHOPS | SIMPKINS,TERRY W. | SPECTRA-PHYSICS | 3310 |
| THE FUTURE OF DATA PROCESSING MANAGEMENT | FROST,RICHARD A. | FUTURE IDEAS, INC. | 3311 |
| THE FUTURE OF STEP BY STEP | KOHOW,MICHEL | TYHLABS CORPORATION | 3312 |

1.3.3 H.P. Philosophy

| | | | |
|---|-------------------|-----------------|------|
| RESPONSE CENTER FROM THE OTHER END OF THE PHONE | SCHOONOVER,CHERIE | HEWLETT PACKARD | 3313 |
|---|-------------------|-----------------|------|

1.3.4 System Security

| | | | |
|---|---------------|----------------------------|------|
| "HELLO" AN UNFRIENDLY GREETING OR AN OFFER OF SEDUCTION ? | HILL,PETER R. | NEGATEC PTY. LTD. | 3314 |
| COMPUTER SECURITY AND LEGAL ISSUES | BLAKE,ISAAC | HEWLETT PACKARD | 3315 |
| SECURITY CONCERNS AND SOLUTIONS | PIRPO,JANINE | OPERATIONS CONTROL SYSTEMS | 3316 |

1.3.5 Focus on Future Technologies

| | | | |
|--|---------------------|------------------------------|------|
| 4GL APPLICATION DEVELOPMENT GUIDELINES | PRALY,MARC | COMGOS INCORPORATED | 3317 |
| ARTIFICIAL INTELLIGENCE - NO LONGER A RESEARCH PROJECT | McEVOY JR.,J. CHASE | McEVOY, COOPER & CO. | 3318 |
| | SPITZER,SUZANNE M. | McEVOY, COOPER & CO. | |
| COMPUTER INTEGRATED MANUFACTURING FOR EXECUTIVES | FLOYD,TERRY H. | BLANKET SOLUTIONS | 3319 |
| FOCUS ON THE FUTURE-THE SEARCH FOR THE SOFTWARE TRANSISTOR | BOSKEY,DAVID | CORPORATE COMPUTER SYSTEMS | 3320 |
| | CHASE,TIM | CORPORATE COMPUTER SYSTEMS | |
| LOGIC PROGRAMMING & EXPERT SYSTEMS | BRAYNAM,SHAWN M. | BRANT COMPUTER SERVICES LTD. | 3321 |

Detroit Interex Conference
Business Proceedings By Author
(HP3000/100)

| | | | |
|------------------------|---|--------------------------------|------|
| ALDINGER,RICK | DISC PERFORMANCE - WHAT IS IT ? | HEWLETT PACKARD | 3102 |
| ATKINSON,TERRY | COMMUNICATING BETWEEN HP'S AND FOREIGN SYSTEMS | | 3111 |
| BARNETT, DONALD | MEETING THE CHALLENGE: AN INSIDE LOOK AT SPECTRUM TESTING | HEWLETT PACKARD | 3128 |
| BLAKE, ISAAC | THE SYSTEM MANAGERS' TOOLBOX | HEWLETT PACKARD | 3110 |
| BLAKE, ISAAC | COMPUTER SECURITY AND LEGAL ISSUES | HEWLETT PACKARD | 3315 |
| BORRESSEN,LARS | INTERPROCESS COMMUNICATION USING MPE MESSAGE FILES | HEWLETT PACKARD | 3112 |
| BOSKEY, DAVID | FOCUS ON THE FUTURE-THE SEARCH FOR THE SOFTWARE TRANSISTOR | CORPORATE COMPUTER SYSTEMS | 3320 |
| BRATHAN, CHRISTOPHER | PEOPLE & MACHINE PERFORMANCE W/SPEEDWARE, POWERHOUSE, RAPID | BRANT COMPUTER SERVICES LTD. | 3206 |
| BRATHAN, SHAWN M. | LOGIC PROGRAMMING & EXPERT SYSTEMS | BRANT COMPUTER SERVICES LTD. | 3321 |
| BRUNO, BENEDICT G. | RIMS, RIMS, RIMS : WHY, WHEN, AND HOW | STR SOFTWARE COMPANY | 3227 |
| BUCHANAN, BUCK | THE SECOND MARKET : BUYING AND SELLING USED HP EQUIPMENT | FIDELITY SYSTEMS | 3305 |
| BURCHETT, RICHARD L. | UTILIZING TELEPHONE USAGE DATA - AUTOMATICALLY | INFOPLOW INTERNATIONAL, INC. | 3235 |
| BUTLER, STEPHEN M. | ADVANCED TRANSACT PROGRAMMING TECHNIQUES | PROBUS INTERNATIONAL INC. | 3204 |
| CARGMONT, LAWRENCE J. | MIGRATION SOLUTIONS FOR MPE/XL | HEWLETT PACKARD | 3130 |
| CASEY-DAVIS, KIMBERLEE | THE KEY TO UNLOCKING PEAK PERFORMANCE | KAISER | 3209 |
| CHASE, TIM | FOCUS ON THE FUTURE-THE SEARCH FOR THE SOFTWARE TRANSISTOR | CORPORATE COMPUTER SYSTEMS | 3320 |
| CHENG, SARA | THE HPIMAGE INTERFACE COMPONENT OF ALLBASE | HEWLETT PACKARD | 3238 |
| COURTNEY, LEE | MPE XL ORGANIZATION AND DIRECTION | HEWLETT PACKARD | 3131 |
| COURTAY, CARY A. | USING THE MPE/XL LINK EDITOR | HEWLETT PACKARD | 3239 |
| COYA, STEPHEN J. | I.25: WHAT TO DO AFTER THE NETWORK IS IN PLACE | NCI DISC | 3125 |
| CRAIG, JACK | PROCESS HANDLING WITH BUSINESS BASIC | BRIDGEWARE | 3106 |
| CROW, BILL | ONLINE INFORMATION SERVICES FOR HP PERSONAL COMPUTER USERS | HEWLETT PACKARD | 3233 |
| DENBRY, PAUL | PERFORMANCE PROGRAMMING WITH HPSQL/V | HEWLETT PACKARD | 3219 |
| DOWLING, JAMES | STRATEGIES FOR EXTENDING THE LIFE OF THE HP 3000 | BOSE CORPORATION | 3107 |
| DRAKE, RONALD | THE ROLE OF THE OPERATOR IN THE DATA CENTER OF THE FUTURE | OPERATIONS CONTROL SYSTEMS | 3114 |
| DUDLEY, KAREN | HEWLETT PACKARD ON COMMUNICATIONS PREMISES WIRING | HEWLETT PACKARD | 3120 |
| EDWARDS, BRUCE | MANAGEMENT - THE FORGOTTEN PART OF M.I.S. | BRIDGE OIL LIMITED | 3308 |
| EHEHART, RICK | MPE VOLUME MANAGEMENT OVERVIEW | HEWLETT PACKARD | 3109 |
| ESCUDES, MANUEL | AUTOCHARGER EXTENDS CARTRIDGE TAPE DRIVE CAPACITY | HEWLETT PACKARD | 3116 |
| ESTES, ROBERTA | HAP (MANUFACTURING AUTOMATION PROTOCOL) | HEWLETT PACKARD | 3123 |
| FARQUHARSON, TAM | 4GL & THE CHANGING ROLE OF THE PROGRAMMER | IMPOCENTRE LTD. | 3213 |
| FIOREAVANTI, PATRICK | THE MINI & THE MICRO-DISTRIBUTED APPLICATION DEVELOPMENT | IMPOCENTRE LTD. | 3217 |
| FIRPO, JAMINE | SECURITY CONCERNS AND SOLUTIONS | OPERATIONS CONTROL SYSTEMS | 3316 |
| FLOTT, TERRY H. | COMPUTER INTEGRATED MANUFACTURING FOR EXECUTIVES | BLANKET SOLUTIONS | 3319 |
| FOSTER, BIRKET | LETS SCOPE IT OUT - FILE TRANSFER BEYOND THE SNEAKER-NET | M.B.FOSTER & ASSOCIATES | 3203 |
| FRIEDRICH, RICHARD | MIGRATING TO THE SERIES 900'S - VARIABLES AFFECT. SYS. PERF. | HEWLETT PACKARD | 3129 |
| FROST, RICHARD A. | THE FUTURE OF SYSTEM DEVELOPMENT | FUTURE IDEAS, INC. | 3208 |
| FROST, RICHARD A. | THE FUTURE OF DATA PROCESSING MANAGEMENT | FUTURE IDEAS, INC. | 3311 |
| GARCIA, I. JAMET | MIGRATION SOLUTIONS FOR MPE/XL | HEWLETT PACKARD | 3130 |
| GREEN, ROBERT M. | ARCHAEOLOGY OF THE HP3000 COMPUTER | ROBELLE CONSULTING LTD. | 3302 |
| GROESSLER, JOERG | IS ONLINE BACKUP POSSIBLE OUTSIDE SPECTRUM ? | JOERG GROESSLER GMBH | 3119 |
| HARMAR, RON | LINKING TO HP SYSTEM DICTIONARY | HEWLETT PACKARD | 3225 |
| HEATER, KAREN | 4GL - THE CONTROVERSY RAGES ON | IMPOCENTRE LTD. | 3201 |
| HECKER, JEFFREY | PROGRAMMING THE NEW GENERATION OF HP COMPS.: A RISC TUTORIAL | AREMS APPLIED ELECTROMAGNETICS | 3132 |
| HELLEBOLD, OLIVIER | WIDE AREA NETWORKING - A CASE STUDY | HEWLETT PACKARD | 3124 |
| HILL, PETER R. | "HELLO" AN UNFRIENDLY GREETING OR AN OFFER OF SEDUCTION ? | MEGATEC PTT. LTD. | 3314 |
| HOLT, WAYNE E. | MEETING THE CHALLENGE: AN INSIDE LOOK AT SPECTRUM TESTING | SOFTWARE RESEARCH NORTHWEST | 3128 |
| HOPMANS, ROSS G. | EXPERT SYSTEM MANAGER FOR THE HP3000 | BRANT COMPUTER SERVICES LTD. | 3113 |
| HOPMANS, ROSS G. | AN EXPERT FINANCIAL PLANNING SYSTEM | BRANT COMPUTER SERVICES LTD. | 3215 |
| HUFFMAN, JACK | BACKUP SYSTEMS, PRESENT AND FUTURE | HEWLETT PACKARD | 3117 |
| HUANG, HEN-JEY | HOW TO DESIGN A CHINESE STYLE NETWORK SYSTEM | | 3303 |
| INDERHILL, BARK | ADOLESCENCE AT THE AGE OF 137: STUDY CHANGE CORPORATE CULTURE | AMFAC DISTRIBUTION CORPORATION | 3301 |
| JAMES, DAVID | IMPACTS OF TECHNOLOGY ON HIGH PERFORMANCE MASS STORAGE | HEWLETT PACKARD | 3104 |
| JORDAN, ARTHUR | APPLICATION PERFORMANCE TUNING - APS/3000 | HEWLETT PACKARD | 3101 |
| KAHINSKI, THOMAS J. | INSURING THE FUTURE OF YOUR DATA BY CONTINGENCY PLANNING | SINGER EDUCATION DIVISION | 3118 |
| KLEINMAN, MITCHELL | METHODS AND PRACTICES OF MIS MANAGEMENT | CONSOLIDATED CAPITAL COMPANIES | 3309 |
| KOHON, MICHEL | A WINDOW INTO THE FUTURE | TYTLABS CORPORATION | 3214 |
| KOHON, MICHEL | THE FUTURE OF STEP BY STEP | TYTLABS CORPORATION | 3312 |
| KONDORFF, AL | MPE FILE SYSTEM OVERVIEW | HEWLETT PACKARD | 3108 |
| KOVALICK, HAY | IS THERE LIFE BESIDES IMAGE ? | HEWLETT PACKARD | 3218 |
| LARSON, ORLAND | RELATIONAL DATABASE : HOW DO WE KNOW WE NEED ONE ? | HEWLETT PACKARD | 3220 |

Detroit Interex Conference
Business Proceedings By Author
(HP3000/100)

| | | | |
|---------------------------|--|--------------------------------|------|
| LAZAR, CLIFFORD W. | POWER TO THE PEOPLE, EXPERIENCES OF A START-UP DP MANAGER | SYSTEMS EXPRESS | 3304 |
| LEVY LEIGHT, BETSY | PROTECTING YOUR SOFTWARE INVESTMENT: AN AUTOMATED APPROACH | OPERATIONS CONTROL SYSTEMS | 3115 |
| LIAT, LIN | THE SPIRIT OF A NEWER SOFTWARE : LI'SPIRIT | SINGAPORE COMPUTER SYSTEMS PTE | 3230 |
| HACKENZIE, DON | AN EXPERT FINANCIAL PLANNING SYSTEM | BRANT COMPUTER SERVICES LTD. | 3215 |
| McBRIDE, BECKY | MIGRATING TO THE SERIES 900'S - VARIABLES AFFECT. SYS. PERF. | HEWLETT PACKARD | 3129 |
| McEVOT JR., J. CHASE | ARTIFICIAL INTELLIGENCE - NO LONGER A RESEARCH PROJECT | McEVOT, COOPER & CO. | 3318 |
| McINTOSH, JC | ADVANCED TRANSACT PROGRAMMING TECHNIQUES | PROBUS INTERNATIONAL INC. | 3204 |
| OUELLETTE, RAYMOND | PROTOTYPING AND SYSTEMS DEVELOPMENT USING 4GL | INFOCENTRE LTD. | 3226 |
| OXFORD, RICHARD | HP2680A. THE MYSTICAL PRINTER, HOW IT WORKS | HCI DISC | 3212 |
| FRALY, MARC | 4GL APPLICATION DEVELOPMENT GUIDELINES | COGMO INCORPORATED | 3317 |
| REHILLARD, ROBERT | INFORMATION CENTERS AROUND 4GL'S | COGMO INCORPORATED | 3202 |
| RICHARDSON, STEVE | HP ADVANCEMET - INTEGRATING NEW PROCESSORS | HEWLETT PACKARD | 3121 |
| ROBINSON, DAVID G. | ENRICHING YOUR POWERHOUSE ENVIRONMENT | ROBINSON, WALLACE & COMPANY | 3205 |
| ROSENBERG, JONATHAN | COMPUTER ASSISTED QUALITY ASSURANCE FOR SOFTWARE DEVELOPMENT | OPERATIONS CONTROL SYSTEMS | 3224 |
| RUTHERFORD, JILL C. | CREATIVE SYSTEM INTEGRATION TO ENHANCE PRODUCTIVITY | BOEING AEROSPACE CO. | 3232 |
| SCHOONOVER, CHERIE | RESPONSE CENTER FROM THE OTHER END OF THE PHONE | HEWLETT PACKARD | 3313 |
| SCHRAH, W.P. | PLANNING INTEGRATED OFFICE SYSTEMS | | 3234 |
| SCOTT, GEORGE B. | IDENTIFYING OPPORTUNITIES FOR PERFORMANCE IMPROVEMENT | ELDEC CORP | 3103 |
| SHAFFER, TIM | ISDN NETWORKING FOR THE OFFICE | HEWLETT PACKARD | 3122 |
| SHONNARD CLARKSON, MARCIA | DATABASE DESIGN IN AN IMPERFECT WORLD | THE UNIVERSITY OF THE SOUTH | 3307 |
| SIELER, STAM | A COMPARISON OF C COMPILERS FOR THE HP3000 | ALLEGRO CONSULTANTS, INC. | 3126 |
| SIMON, RICK | COMMERCIAL SPECTRUM PROGRESS REPORT | HEWLETT PACKARD | 3127 |
| SIMPKINS, TERRY W. | STRATEGIC PLANNING IN SMALL HIS SHOPS | SPECTRA-PHYSICS | 3310 |
| SINCLAIR, JIM | MIGRATING POWERHOUSE APPLICATIONS TO NEW MACHINE ENVIRONMENT | COGMO INCORPORATED | 3237 |
| SOLLAND, LEIGH | THE SYSTEM LIFE CYCLE IN THE 4GL ENVIRONMENT | COGMO INCORPORATED | 3210 |
| SPENCE, STEVEN J. | MIGRATING COBOL PROGRAMS TO SPECTRUM: A BATTLE OR A BREEZE ? | HEWLETT PACKARD | 3236 |
| SPITZER, SUZANNE M. | ARTIFICIAL INTELLIGENCE - NO LONGER A RESEARCH PROJECT | McEVOT, COOPER & CO. | 3318 |
| TASHENBERG, C. BRADLEY | TRENDS IN IMAGE | BRADMARK COMPUTER SYSTEMS | 3223 |
| TEMPLE, YVONNE | GRAPHICS IN AN ORGANIZATION | HEWLETT PACKARD | 3211 |
| TOBACK, BRUCE | SOFTWARE DESIGN FOR LONG-TERM RELIABILITY & MAINTAINABILITY | OPT. INC. | 3228 |
| TRASKO, MARK S. | THE FUTURE OF DATABASE TECHNOLOGY | DYNAMIC INFORMATION SYSTEMS | 3222 |
| VAN VALKENBURGH, R.E. | IMPROVING YOUR PERFORMANCE | AMPEX CORPORATION | 3105 |
| VIRGILIO, LESLIE A. | INSURING THE FUTURE OF YOUR DATA BY CONTINGENCY PLANNING | SINGER EDUCATION DIVISION | 3118 |
| VOLOKH, EUGENE | RELATIONAL DATABASES VS. IMAGE: WHAT THE FUSS IS ALL ABOUT | VEESOFT, INC. | 3221 |
| WALLACE, MARK | THE TOOLS OF STRUCTURED ANALYSIS - A TUTORIAL | ROBINSON, WALLACE & COMPANY | 3231 |
| WALLACE, MARK | HOW TO DEVELOP NEW APPLICATIONS - A STRATEGY | ROBINSON, WALLACE & COMPANY | 3216 |
| WARZECHA, CHARLES E. | PERSONAL COMPUTERS SOLVE 4GL PROBLEMS | GATEWAY SYSTEMS CORPORATION | 3207 |
| WHITE, FRED | THE FOURTH BEAR OF IMAGE | ADAGER | 3229 |
| WISEMAN, D.B. | WHAT HP DIDN'T TELL YOU WHEN YOU BOUGHT YOUR HP3000 | SYSTEM SOFTWARE LIMITED | 3306 |

APPLICATION PERFORMANCE TUNING - APS/3000

Arthur Jordan
Hewlett-Packard Company
Computer Systems Division

I. INTRODUCTION

1. Overview

Application Program Sampler/3000 (APS/3000) is an interactive performance tool that allows you to "fine tune" HP3000 programs. APS/3000 monitors the execution of a user specified program and produces histograms that depict the relative amount of time spent executing program statements. After examining the histograms, you can optimize time-consuming program statements and then repeat the procedure until you achieve the level of efficiency you desire.

2. APS/3000 Features

APS/3000 allows you to:

- o Measure performance without having to modify your source code.
- o Monitor the execution of a single program or multiple executions of one or more shared programs.
- o Distinguish between direct CPU utilization (i.e., CPU time spent executing user program statements) and indirect CPU utilization (i.e., CPU time executing user SL code or system SL code on behalf of the user program).
- o Estimate the rate of transfer of control between segments (for segmentation purposes).
- o Determine wait times (i.e., time when the process is not active - either waiting for a system function or user think time) of program statements in order to estimate turnaround time.
- o Analyze the log file created during the data collection process on another system without any loss of functionality.

3. APS/3000 Requirements

APS/3000 can analyze programs written in COBOLII, SPL, FORTRAN, PASCAL, or BASIC. A PB-location listing is needed to correlate the APS/3000 histograms with the statements in a program. Either a PMAP listing is needed, or the program must be PREPped using the FPMAP option.

4. APS/3000 Modules

APS/3000 consists of three modules. The user interacts with each module through a hierarchical series of menus. The modules are: SAMPLER, DISPLAY, and ANALYZER.

SAMPLER is the data collection module. It monitors the execution of one or more programs and stores the samples in a log file.

DISPLAY provides real time histograms during the data collection process of the measurement as it progresses.

ANALYZER reduces the data gathered by SAMPLER. The ANALYZER menus allow you to analyze the data in increasingly finer steps. You can examine: CPU time for all sampled programs and processes, CPU time for segments within a given program, and, procedure and address ranges within a given segment.

5. When to use APS/3000

The information produced by APS/3000 is useful throughout a program's lifecycle:

- o Before you release the program to the user for the first time.
- o After you have added enhancements to the program.
- o When you need to improve poor response times.
- o When contention for shared resources is degrading system performance

After briefly describing the MPE architectural features that underlie the operation of APS/3000, the procedures for "tuning" a program are described. The paper concludes with a case study which shows how to use APS/3000 to collect data and analyze a COBOLII program step-by-step.

II. MPE AND APS/3000 RELATED TOPICS

1. Process Identification Number – PIN

A process is the unique execution of a program by one user. MPE identifies each process by a unique number called a process identification number (or PIN). During process creation, a free PIN is allocated to the process. The PIN is associated with the process until it terminates. Upon process termination, the PIN is freed for use by another process.

2. Programs and Segmented Libraries

A program can reference procedures from three different sources: the group SL, the pub SL, or the system SL. A program performs a unique function (e.g. a marketing software application). Group and pub SLs are managed by the user and contain procedures that perform a specific function for a program. The system SL contains procedures (termed *intrinsic*s) that are called by programs or by user SL procedures.

3. Segments

Programs and SLs are made up of segments. A process can have up to 255 segments (in any combination of program segments and user/system SL segments). Program files can contain up to 63 segments.

The COBOLII reserved word *SECTION* defines the segment in which the code following it will reside. The syntax is "seg-name SECTION #", where *seg-name* is a user-defined name for the segment. The # defines the COBOLII internal segment number where the code will reside.

The COBOLII source statement "MAIN-LINE-SECTION SECTION 0." is translated to "MAINLINESECT00". The COBOLII internal segment name is "MAINLINESECT" and the internal segment number is "00". The programs actual segment numbers produced by the Segmenter (via the *PREP* command) do not match the COBOLII internal segment numbers.

4. Shared Clock Interface

The Shared Clock Interface is a set of system procedures that allows APS/3000 to interrupt the operating system at a given fixed interval. The user can select a sampling interval from 5 to 1000 milliseconds. The interrupt is a high-priority event that can preempt most non-I/O drivers and active programs.

5. Stack Markers

When a subprogram or SL procedure is called, MPE uses a stack marker to save the origin of the call. The stack marker contains the segment number and PB-location of the caller. Stack markers are chained together to reflect the history of procedure calls up to the present.

6. Collecting Direct, Indirect, and Wait Samples

Typically, you specify the program or programs you want to study and instruct the *SAMPLER* module to perform a new measurement. *SAMPLER* interrupts processing at each occurrence of the sampling interval. First, *SAMPLER* determines if the interrupted program is the program (or one of the programs) you specified. If so, *SAMPLER* reads the chain of stack markers. If the last stack marker describes a segment in the program, *SAMPLER* records it in the log file as a direct sample.

If the last stack marker is for a SL segment (i.e. group SL, pub SL, or system SL), *SAMPLER* records its segment number and PB-location. Then, *SAMPLER* begins tracing back through

successive stack markers until it finds a program segment. The data is then recorded to the log file as an indirect sample.

SAMPLER records Direct CPU Cost, Indirect CPU Cost, and Wait time samples. Direct CPU Cost represents execution time of your program. Indirect CPU Cost represents execution of SL procedures on behalf of a program. Wait time represents I/O delays due to hardware limitations, processing delays due to system load factors, and user think times. For example, a program initiates a read to a terminal and the user has to wait some time, say a minute, before they respond. This minute is accumulated as wait time. Technically, Wait Time is the accumulated total number of sampler interrupts during which the program being studied was not active. The example below clarifies the distinction between the three measurement types.

```
MOVE "ENTER YOUR NAME:" TO WRITE-LINE.
DISPLAY WRITE-LINE.
ACCEPT NAME.
```

Direct CPU Cost is incurred when "ENTER YOUR NAME:" is moved to WRITE-LINE. Indirect Cost is incurred when the system executes the DISPLAY WRITE-LINE. Wait time will be accumulated if the user waits some amount of time before entering their name.

ANALYZER constructs histograms that depict these costs. ANALYZER reports Direct Cost histograms, Direct + Indirect Cost histograms, and Turnaround time (Direct + Indirect + Wait Time) histograms. Turnaround histograms depict the sum of the direct, indirect, and the wait time for a given execution of the program being studied.

7. VERBS map

ANALYZER's histograms depict PB-locations within a segment. The COBOLII compiler reads each statement in your source file and then generates the machine instructions needed to accomplish the statement.

The VERBS map (produced by the COBOLII compiler's VERBS directive), lists the beginning PB-location of each VERB in the program. The example below is a portion of a compiler listing and the matching listing for the VERBS map.

```
LINE #
00045      005000 C100-READ-DATA-FILE.
00046      005100      READ DATA-FILE AT END MOVE "Y" TO EOF.
00047      005200      MOVE DATA-FILE-REC TO TERM-LINE.
00048      005300      MOVE SPACES TO WRITE-LINE.
00049      005400 C100-EXIT.
```

```
LINE #    PB-LOC  #    PROCEDURE NAME/VERB
00045     000036      C100-READ-DATA-FILE
00046     000036      READ
00046     000054      MOVE
00047     000057      MOVE
00048     000065      MOVE
00049     000076      C100-EXIT
```

The "LINE #" in the VERBS map corresponds to the "LINE #" in the compiler listing. The "PB-LOC" is the starting PB-location for each VERB.

<label>: Defines the entity the histogram is describing. This could be a PIN, program name, segment name, or an address range.

<bar-field>: Describes the relative costs for the entities. The <bar-field> can be described using a "D" for direct, an "I" for indirect, and a "W" for wait times.

CNT: Contains the number of samples described in the bar-field. The CNT field is only available on the hardcopy listings of histograms.

%: Field represents the percentage of samples for the entity to the total number of samples for the histogram.

%CUM: Defines the cumulative percent.

ANALYZER displays three different types of histograms. They are:

1. Direct Histogram: Displays the relative CPU cost for the program statements.
2. Indirect Histograms: Combines the Direct CPU cost with the Indirect CPU cost.
3. Wait Histograms: Combines Direct CPU cost, Indirect CPU cost, with the Wait time.

Histograms at the "Procedure and Address Level" for different segments cannot be compared with one another to understand which segment consumed the most CPU time. Segment histograms are displayed in decreasing CPU utilization.

III. HOW TO TUNE A PROGRAM

Tuning a program requires that a series of steps be followed. They are:

1. Define a program test procedure
2. State current performance and objectives
3. Generate new listings and program file
4. Collect the data
5. Analyze log file
6. Make source code modifications
7. Repeat steps 3 through 6 until desired performance objectives have been reached

STEP 1: Define a program test procedure

You must control the environment to ensure that the improved performance is the result of modifying the source code and not random variation. The ability to duplicate the exact same test is critical in tuning a program. Four important points to follow in defining a reproducible test are:

1. Use a method of measuring the CPU time that is consistent for each test. For an interactive program, this might be the duration between a carriage return and when the screen is repainted with the requested data. This could be measured by a stop watch. Better results would be obtained by displaying time stamps after the carriage return and before the requested data is displayed.

For batch programs, the measurement would be the time it takes the program to execute.

2. If the program modifies data in data file(s), those files must be restored to their original state prior to the re-execution of the test.

3. The test must be executed in a standalone environment (i.e., nothing else is running on the system). Otherwise, another program could execute during the test and thus spuriously inflate execution time for the test.

4. If the program is interactive, the same screen data must be entered for each re-execution of the test.

Once a test has been defined it should be run several times to insure it reproduces the same execution times for each test.

STEP 2: State current performance and objectives

Before any performance enhancements are made to the program, a base set of performance statistics must be documented. After each execution of the test, the performance statistics should be documented and compared with the last test. Sometimes "performance enhancements" actually degrade performance rather than improve it.

Step 1 should provide the necessary information to state the current performance statistics (i.e. what is the measured response time of the program?). Any variance between successive runs of the test should be documented.

The desired performance objectives should also be understood. The objectives are statements that describe what the performance of the program should be. For example:

- o The interactive program must have a response time of under 15 seconds.
- o The batch program must complete in under 4 hours.

It is important to document these objectives so you know when they have been reached. Otherwise, performance tuning can go on forever.

STEP 3: Generate new listings and program file

In order to correlate the profiles produced by the ANALYZER module, two listings must be available: the compiled code and the VERBS map (this assumes that the FPMAP option was used when the program was PREPPed).

Before tuning, generate a compile listing with the VERBS map and a new program file. Don't assume that the listings available match the program file being analyzed (Only a small change was made. To save paper, no new listings were generated). It is very important that the listings exactly match the program being studied. The profiles from the ANALYZER might be 10 to 100 PB-locations different depending upon modifications made. If the code locations do not match, the wrong source code will be modified when making performance enhancements!

STEP 4: Collect the data

The SAMPLER module is used to collect data. It is entered by selecting "NM" (New Measurement) on the main screen in APS/3000. This will display the Measurement Option Commands. Three different types of measurement options are available. They are:

1. RM: This type of measurement is used when only one program is to be analyzed. Data is collected for one program executed by one process. If it is an online program, interaction with the program will occur on the terminal. This type of measurement allows the user to display the Program/Process Level histograms through the Procedure/Address Level histograms.
2. AM: Is used when analyzing one or more programs. Data will be collected for all processes executing the program(s) the user selects. Up to ten programs can be monitored. This type of measurement allows the user to display the Program/Process Level histograms through the Procedure/Address Level histograms.

3. AL: When the entire system is to be measured, this option should be used. Data is collected from all programs executed by all users. Using this measurement option, only the Program/Process Level through Segment Level histograms are available.

If only one program is to be analyzed, the RM measurement option should be used. If a program has several sub-programs executing and communicating together, the AM measurement option should be used. To analyze one program executed by several users, AM mode should be used. AL measurement option is used to review which system segments are being used (not for tuning programs).

STEP 5: Analyze log file

The ANALYZER module is used to analyze a log file. ANALYZER can be entered through the main screen of APS/3000 using the "RE" (Replay and Analysis of log file) command.

Two general rules exist that help in analyzing a log file. They are:

1. Eighty percent of the CPU time is spent in 20 percent of the program. If this is true, the strategy behind analyzing a log file is to find where the program is spending 80 percent of its time and enhance that portion.
2. Eighty percent of the time, a program is executing system code (e. g., calling intrinsics).

Again, these are general rules which are fairly accurate. They help establish a direction and confirm where performance enhancements could be made.

Analyzing a program is accomplished by repeating a series of steps over and over again, until all of the high CPU utilized areas have been analyzed. The process is:

1. Review the segment level histograms. Locate the segment that consumed the most CPU time.
2. Go to the procedure / address level histograms (reviewing the segment found in step 1). Find the address range that consumed the most CPU time.
3. Trace that to the compiled code listing.
4. Find possible solutions to enhance that portion of code so it will be more efficient. This usually means taking out that section of code, rewriting it, or leaving it in because it cannot be changed.
5. Go to step 2, find the next highest CPU utilized address range. Repeat until all of the high CPU utilized address ranges have been exhausted for that segment.
6. Go to step 1 and find the segment that consumed the next highest CPU utilization.

This process is performed for all of the direct CPU histograms and the indirect CPU histograms.

Direct histograms show which code should be revised to make the program more efficient. The indirect histograms point to calls made to SL procedures that needs to be reviewed for performance enhancements. If the SL is user-managed (group SL or pub SL), the call can be taken out of the program, the procedure in the SL can be enhanced, or it can be left

unchanged. If the SL procedure is an intrinsic, ask yourself, "can the intrinsic call be deleted? If the intrinsic is file-system related, can the file structure be changed for faster turn-around time on the call?"

STEP 6: Make source code modifications

Two strategies exist for modifying the source code with the changes found in step 5. All of the changes can be done at once or they can be done one at a time. Performing one change at a time offers the satisfaction of knowing if the change increased performance, had little effect, or degraded performance. The source is modified and recompiled, new listings are generated, and the test is re-run again. Although this is time consuming, it is the best way to install performance enhancements.

Summary

Tuning a program can be a never-ending process. There is always one more area of code in the program that is devouring CPU cycles. At some point, the increase in performance does not warrant the analysis and code changes. Hopefully, you have met the performance objectives before that point is reached.

This process not only improves the performance of existing programs, but also shows you what consumes CPU time on the HP3000. This will assist you in designing new programs that are more efficient.

APS/3000 not only points out where in the program CPU time is being spent (which is a small part of the overall execution time), but also what calls to SL procedures (i.e., calls to intrinsics) consume CPU time. This is also valuable information. It can assist in pointing out which file structures are consuming a lot of CPU time. Hopefully, this will lead to tuning those file structures.

After analyzing a program, you may decide that no performance enhancements can be made because all of the code is performing necessary functions. In this case, there is the satisfaction of knowing that the program is running as efficiently as possible.

All performance problems can be solved by purchasing more hardware. An intelligent decision will be based on whether a software or hardware solution is most cost effective.

Before purchasing more hardware, evaluate the possibility and cost of tuning the programs that consume the most CPU time. Then, evaluate the system performance (Maybe you only need to purchase more memory). Finally, consider an upgrade or an additional CPU.

APS/3000 does not have to be used only in a reactive mode. In a proactive mode it can ensure that a program is using system resources efficiently and that the user is getting the fastest response time possible.

IV. CASE STUDY

1. Introduction

The case study will walk through the data collection and analysis phase of tuning a program. The program analyzed is a short COBOLII program called TESTPROG. It was designed to show the three main functions of APS/3000. These are: To determine where the most amount of CPU time was spent executing program code (direct). To determine where in the program the most amount of CPU time was spent executing called procedures in SLs (indirect). How to interpret wait times.

The following sections contain the compile listing, VERBS map listing, and a menu-by-menu walk through of data collection and data reduction. User input will be in bold upper case letters. A **(RETURN)** is a carriage return and implies a YES response in APS/3000.

2. The compile and VERBS map listings

The TESTPROG compiler listing and VERBS map listing is provided on the following pages. The PMAP listing is not needed because the FPMAP option was used when PREPPing TESTPROG.

Browse through the compiler listing to understand what the program does. Two points about the compiler listing, they are:

1. The VERBS map compiler directive is found in the compiler listing line number 4 (these are the numbers to the far left of the listing).
2. The program is segmented only to show how the COBOLII compiler translates the source code segment names to the COBOLII internal segment names. APS/3000 refers to the segments by the COBOLII internal segment names. To see how they get translated, review the SECTION verb in the compile listing and then find the corresponding line number in the VERBS map listing to find the COBOLII compilers internal name.

TESTPROG contains one main loop. Within this loop it:

- o Opens a data file
- o Reads 100 records from the data file
- o Closes the data file
- o Moves spaces to a buffer
- o Prompts the user to re-execute the loop

```

00004      001000$CONTROL USLINIT,VERBS      <- Point 1
00006      001100 IDENTIFICATION DIVISION.
00007      001200 PROGRAM-ID.          APS3000-TEST.
00008      001300 ENVIRONMENT DIVISION.
00009      001400 INPUT-OUTPUT SECTION.
00010      001500 FILE-CONTROL.
00011      001600      SELECT DATA-FILE ASSIGN TO "DATAFILE".
00012      001700 DATA DIVISION.
00013      001800 FILE SECTION.
00014      001900
00015      002000 FD DATA-FILE
00016      002100      RECORD CONTAINS 80 CHARACTERS.
00017      002200 01 DATA-FILE-REC          PIC X(80).
00018      002300
00019      002400 WORKING-STORAGE SECTION.
00020      002500 77 Y-N                      PIC X.
00021      002600 77 EOF                      PIC X.
00022      002700 01 WRITE-LINE.
00023      002800      05 TERM-LINE          PIC X(80).
00024      002900      05 TERM-LINE1        PIC X(1920).
00025      003000
00026      003100 PROCEDURE DIVISION.
00027      003200 MAIN-LINE-SECTION SECTION 0.      <- Point 2
00028      003300 A100-MAIN-LINE.
00029      003400      PERFORM B100-OPEN-DATA-FILE THRU B100-EXIT.
00030      003500      PERFORM C100-READ-DATA-FILE 100 TIMES.
00031      003600      PERFORM D100-CLOSE-DATA-FILE THRU D100-EXIT.
00032      003700      PERFORM E100-CLEAR-SMALL-BUF THRU E100-EXIT.
00033      003800      MOVE SPACES TO WRITE-LINE.
00034      003900      DISPLAY "PROCESS A100-MAIN-LINE AGAIN".
00035      004000      ACCEPT Y-N. IF Y-N = "Y" GO TO A100-MAIN-LINE.
00036      004100      STOP RUN.
00037      004200 A100-EXIT.
00038      004300
00039      004400 WORKING-SECTION SECTION 1.      <- Point 2
00041      004600 B100-OPEN-DATA-FILE.
00042      004700      OPEN INPUT DATA-FILE.
00043      004800 B100-EXIT.
00045      005000 C100-READ-DATA-FILE.
00046      005100      READ DATA-FILE AT END MOVE "Y" TO EOF.
00047      005200      MOVE DATA-FILE-REC TO TERM-LINE.
00048      005300      MOVE SPACES TO WRITE-LINE.
00049      005400 C100-EXIT.
00051      005600 D100-CLOSE-DATA-FILE.
00052      005700      CLOSE DATA-FILE.
00053      005800 D100-EXIT.
00055      006000 E100-CLEAR-SMALL-BUF.
00056      006100      MOVE SPACES TO WRITE-LINE.
00057      006200      MOVE SPACES TO WRITE-LINE.
00058      006300 E100-EXIT.

```


The VERBS map listing shows the starting PB-location of each VERB in the program. The "LINE #" is the compiler listing line number. The "PB-LOC" is the starting PB-location of each verb. The "PROCEDURE NAME" is the name of each segment defined in the program. The "INTERNAL NAME" is the COBOL translated segment name in the SECTION verb.

| LINE # | PB-LOC # | PROCEDURE NAME/VERB | INTERNAL NAME |
|--------|----------|-------------------------|--------------------------|
| 00027 | 000003 | 0 MAIN-LINE-SECTION | MAINLINESECT00'← Point 2 |
| 00028 | 000003 | A100-MAIN-LINE | |
| 00029 | 000003 | PERFORM | |
| 00030 | 000007 | PERFORM | |
| 00031 | 000021 | PERFORM | |
| 00032 | 000026 | PERFORM | |
| 00033 | 000032 | MOVE | |
| 00034 | 000041 | DISPLAY | |
| 00035 | 000100 | ACCEPT | |
| 00035 | 000100 | IF | |
| 00035 | 000110 | GO TO | |
| 00036 | 000116 | STOP | |
| 00037 | 000123 | A100-EXIT | |
| 00039 | 000003 | WORKING-SECTION | WORKINGSECTI01'← Point 2 |
| 00041 | 000003 | B100-OPEN-DATA-FILE | |
| 00042 | 000003 | OPEN | |
| 00043 | 000034 | 5 B100-EXIT | |
| 00045 | 000036 | C100-READ-DATA-FILE | |
| 00046 | 000036 | READ | |
| 00046 | 000054 | MOVE | |
| 00047 | 000057 | MOVE | |
| 00048 | 000065 | MOVE | |
| 00049 | 000076 | C100-EXIT | |
| 00051 | 000076 | D100-CLOSE-DATA-FILE | |
| 00052 | 000076 | CLOSE | |
| 00053 | 000103 | D100-EXIT | |
| 00055 | 000105 | 10 E100-CLEAR-SMALL-BUF | |
| 00056 | 000105 | MOVE | |
| 00057 | 000117 | MOVE | |
| 00058 | 000126 | E100-EXIT | |

3. Collecting the data

To execute APS/3000, run SAMPLER.PUB.SYS. The main menu of the SAMPLER module will be displayed.

```
(C) HEWLETT/PACKARD 1980 HP32180A.01.07 (A.01.07) WED, JUN 11, 1986
Application Program SAMPLER/3000
```

This is a software tool for tuning application and system programs. The SAMPLER module can invoke, monitor, and log measurements of one or more programs. The DISPLAY module offers a real-time view of the data currently being observed by SAMPLER. ANALYZER reads logged data and produces histograms that break down the relative CPU cost of program file names, process names, segment names, procedure names, or address ranges relative to the beginning of a segment or procedure. The programmer then decides which program areas to optimize.

SAMPLER's Function Selection Commands (Enter H for Help):

^NM: New Measurement (SAMPLER)

RE: Replay and Analysis of Log File (ANALYZER)

| | | |
|---------------|----------------|---------------------------|
| A: Automatic^ | M: Menu | CL: Clear Screen** |
| E: End | QU: Quiet | RL: Roll Up |
| EX: Exit | VQ: Very Quiet | HC: Hard Copy only |
| H: Help | VB: Verbose** | TD: Terminal Display only |
| T: Top | | TH: Terminal & Hard Copy |

SAMPLER-> **NM**

An "NM" is entered to initiate a new measurement. The Measurement Option menu will be displayed next.

SAMPLER has the capability to a) invoke and monitor the execution of a specific program, b) allocate and monitor the shared execution of one or more programs or c) monitor all active programs including operating system functions.

Select one of these options to continue. Enter H for HELP.

Measurement Option Commands:

^RM: Run and Monitor an Application Program
AM: Allocate and Monitor a Shared Program(s)
AL: Monitor All Active Programs Including MPE

| | | |
|---------------|----------------|---------------------------|
| A: Automatic^ | M: Menu | CL: Clear Screen** |
| E: End | QU: Quiet | RL: Roll Up |
| EX: Exit | VQ: Very Quiet | HC: Hard Copy only |
| H: Help | VB: Verbose** | TD: Terminal Display only |
| T: Top | | TH: Terminal & Hard Copy |

SAMPLER-> **RM**

The measurement option "RM" is selected because the execution of TESTPROG by one user will give the controlled environment needed.

Enter a program name and other optional parameters to execute it. If the normal execution of your program requires any special :FILE equations, break from SAMPLER and define them (if not already done) and then resume SAMPLER.

Program Name: **TESTPROG**
 Primary Entry Point? **(RETURN)**
 Libsearch S? < G | P | S | YES | [RET] for S > **(RETURN)**
 End of Parameters? **(RETURN)**

Program Name: The name of the program to be analyzed should be entered. TESTPROG is the name of the COBOLII program that will be analyzed.

Primary Entry Point?: Programs can contain alternate entry points. The default is the primary entry point. See the CREATE intrinsic for more information.

Libsearch S?: If a group SL is used enter "G". For programs that reference a pub SL, enter a "P". Otherwise enter a "carriage return".

End of Parameters?: If no other parameters exist enter a "carriage return". If other parameters are required enter a "yes". This will initiate a series of questions to the user (e.g., XDATA, INFO string, etc).

SAMPLER is now set up to run and monitor the specified program. Enter command BL to begin measurement and log samples. Measurement parameters can be changed in the dialogue that follows.

Measurement Activation Commands:

^BL: Begin and Log Measurement
BE: Begin Measurement (No logging)
ST: Stop Measurement (& Close Log File if Any)
PU: Pause Sampling
RS: Resume Sampling
SU n: Set Auto Update Delay to n Seconds (e.g. SU 20)

VR: Verify Measurement Parameters
SH: Show Statistics
OD: Activate On-Line DISPLAY

| | | |
|-----------------------|------------------------|------------------------------------|
| A : Automatic^ | M : Menu | CL : Clear Screen** |
| E : End | QU : Quiet | RL : Roll Up |
| EX : Exit | VQ : Very Quiet | HC : Hard Copy only |
| H : Help | VB : Verbose** | TD : Terminal Display only |
| T : Top | | TH : Terminal & Hard Copy** |

SAMPLER-> **BL**

Selection "BL" starts the measurement process and writes the samples to a log file.

First, to start the measurement, SAMPLER prompts for a series of questions. Once answered, SAMPLER invokes TESTPROG as a son process. It executes on the same terminal as APS/3000 is executing. When TESTPROG has completed, SAMPLER displays some measurement statistics and prompts to go on to analyze the log file. At this point data collection is complete.

For the following prompts enter YES or Carriage Return [RET] to confirm the default values. Enter NO (or a new value) in order to override defaults.

Sampling Interval 10 Milliseconds? 5

For Direct CPU Time Measurement
All program segments and
Pin %235 will be monitored.

Do You Want Indirect CPU Time (and Wait) Measurement Too? (RETURN)

"Measurement Title" -> "**Collection - PASS 1**"
Log File SAMPLOG? (RETURN)

You Will Need No More Than 66 Records of Log File Per Minute of Sampling.

Log File Size 1023 Records [RET]= YES ? (RETURN)

Measurement Parameters:

Title: APS/3000 DEMO - PASS 1
Measurement Begin: WED, JUN 11, 1986, 7:22 PM
Measurement End:
Program(s): TESTPROG
Measurement Option: RUN & MONITOR A PROGRAM (RM Command)
Current Activity: SAMPLING & LOGGING
Sampling Interval: 5 MILLISECONDS
Process (PINs): %235
Measurement Type: DIRECT AND INDIRECT CPU TIME
Log File: SAMPLOG.JORDAN.SS

All Right? (RETURN)

Sampling Interval: SAMPLER automatically determines the correct sampling interval based on HP3000 series. Because the program is very short, a sample interval of 5 was used.

Do You Want Indirect CPU Time Measurement Too?: We want to study the indirect cost and wait times.

Measurement Title: Measurement titles should indicate which pass of a test is being run.

Log File SAMPLOG: Previous log files should be saved to allow comparisons. If no name is entered when prompted for "Log file SAMPLOG", SAMPLER defaults to a log file named SAMPLOG. If you enter "NO", SAMPLER will prompt for a new log file name.

Log File Size 1023 Records: SAMPLER estimates how large to make the log file and uses a worst case of 66 records per minute. This is very high. Such a test would execute for one minute and use only 3 records of the log file. When in doubt, make the log file larger than needed. Once the log file is full, SAMPLER will continue measuring but samples will not be recorded to the log file.

SAMPLER/3000 measurement is in progress. WED, JUN 11, 1986

PROCESS A100-MAIN-LINE AGAIN
Y
PROCESS A100-MAIN-LINE AGAIN
Y
PROCESS A100-MAIN-LINE AGAIN
Y
PROCESS A100-MAIN-LINE AGAIN
Y
PROCESS A100-MAIN-LINE AGAIN
Y
PROCESS A100-MAIN-LINE AGAIN
N

Measurement Completed.

Measurement Stopped.

403 Samples 4839 Interrupts WED, JUN 11, 1986, 7:23 PM
3 Records in Log File

Please wait. Run environment is being logged.

PMAP data was extracted from file: TESTPROG

Do You Want to Analyze Log File Now?

Data collection is finished!

The main loop of the TESTPROG executed 6 times before an "N" was entered to terminate the program. At each TESTPROG prompt (i.e., PROCESS A100-MAIN-LINE AGAIN) a delay of 3 seconds for user wait time was used before entering a response.

The log file SAMPLOG contains 403 samples. This is a combination of direct and indirect samples. Wait times are part of the indirect samples.

4. Data Reduction

The ANALYZER module has three levels of histograms: Program and Process Level, Segment Level, and Procedure and Address Level. We will display histograms at all three levels to diagnose where the Direct CPU Cost, Indirect CPU Cost, and wait time were consumed.

First, ANALYZER prompts for the log file name and a subtitle. The characteristics of the log file are reported and then the Program and Process Level menu is displayed.

Analysis of SAMPLER's Log File

ANALYZER processes the log file generated by SAMPLER and produces CPU time profiles of sampled program(s). Now specify the log file name:

Log File SAMPLOG ? (Yes/No) **RETURN**

Enter "Subtitle", [RET] if none -> **"PASS 1 ANALYSIS"**

****** Processing of log file begun. Please wait. ******

```

Measurement Title:  APS/3000 DEMO - PASS 1
      Begin:  WED, JUN 11, 1986,  7:22 PM
      End:    WED, JUN 11, 1986,  7:23 PM
Analysis Subtitle:  PASS 1 ANALYSIS OF APS/3000 DEMO
Program(s):        TESTPROG.JORDAN.SS
Log File:          SAMPLOG      Version:  A.01.07
Measurement Option:  RUN & MONITOR A PROGRAM (RM Command)
Logfile obtained from system with mapping microcode
Measurement Type:   DIRECT AND INDIRECT CPU TIME
Sampling Interval:  5 MILLISECONDS
Number of Samples:  403
      OnICS Samples:  7
Machine:           SERIES 64
Memory Size:       4096 Kilo Words
MPE Version:       MPE: G.01.05  BASE: G.01.05
  
```

Log File SAMPLOG?: If the name of the log file is different than the logfile you want to analyze, enter a "no". ANALYZER will then prompt for the correct log file name.

Enter Subtitle: Data reduction might require several passes through the log file. The subtitle should describe the pass level.

Segment Level

Program **TESTPROG.JORDAN.SS** will be analyzed.

Segment Level Commands:

- ^DA:** Direct CPU Utilization by All Segments
- DS:** Direct CPU Utilization by System Segments
- DU:** Direct CPU Utilization by User Segments
- IU:** Direct & Indirect Utilization by User Segments
- TR:** Segment Transition Statistics
- IS:** Indirect Cost of Individual User Segments
- ST:** SL Segment Transitions (AM measurement mode only)
- PU:** Segment Transition by Procedures
- N:** Next - Move to Procedure and Address Level

LM >= n: Set Bar Threshold >= n%
 LM > n: Set Bar Threshold > n% (LM > 0% is default)

- | | | |
|----------------------|-----------------------|-----------------------------------|
| A: Automatic^ | M: Menu | CL: Clear Screen** |
| E: End | QU: Quiet | RL: Roll Up |
| EX: Exit | VQ: Very Quiet | HC: Hard Copy only |
| H: Help | VB: Verbose** | TD: Terminal Display only |
| T: Top | | TH: Terminal & Hard Copy** |

ANALYZER-> **DA**

To get an overall picture of the CPU cost attributed to user and system segments, enter the "DA" command (Direct CPU Utilization by All Segments).

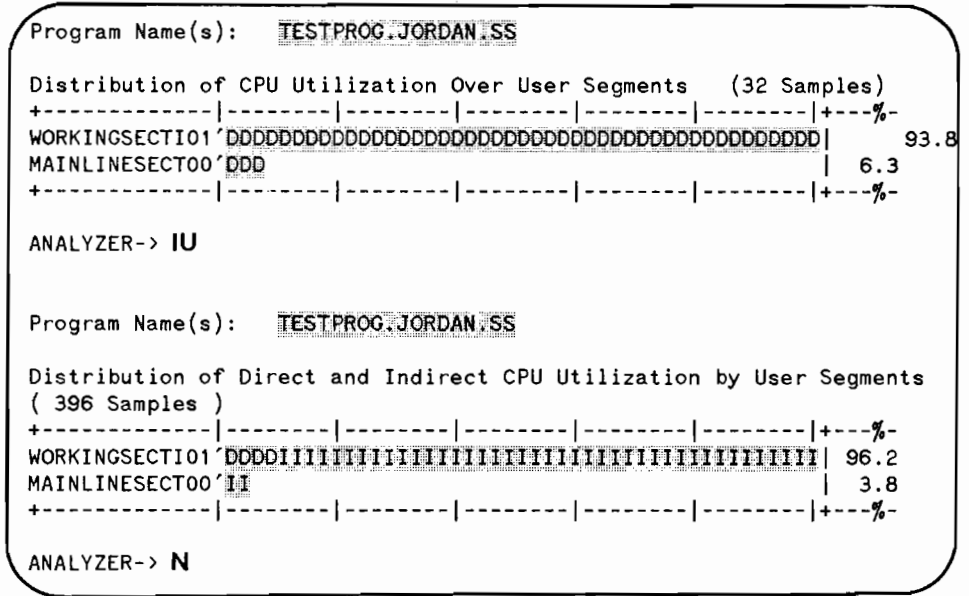
Program Name(s): **TESTPROG.JORDAN.SS**

Distribution of Direct CPU Utilization Over All Segments
(403 Samples - Including OnICS Samples)

| Segment Name | Utilization (%) |
|----------------|-----------------|
| FILESYS6A | 1.0 |
| FILESYS4 | 1.2 |
| ALLOCUTIL | .2 |
| HARDRES | 5.2 |
| KERNELC | 5.0 |
| MISCSEGC | 4.2 |
| FILESYS1A | 34.2 |
| CACHESEG | 33.0 |
| HIOMDSC2 | .7 |
| WORKINGSECTI01 | 7.4 |
| MAINLINESECT00 | .5 |
| COBLIB18 | 1.5 |

ANALYZER-> **DU**

The last two histograms to review at the Segment Level are the Direct CPU Utilization by User Segments and Direct & Indirect Utilization by User Segments. These two histograms show which user segments were active and what the direct and indirect CPU costs were.



What can we say about TESTPROG so far:

- o There are two segments in TESTPROG that consumed CPU time. They are "WORKINGSECTI01" AND "MAINLINESECT00".
- o Most of the time was spent in SL code. There were 32 direct samples and 364 indirect samples (subtract the number of direct samples from the number of direct + indirect samples, or 396 - 32 = 364).
- o Segment "WORKINGSECTI01" consumed more CPU time than segment "MAINLINESECT00".

SUMMARY: The largest improvement to TESTPROG would be to cut down the indirect CPU cost in the segment "WORKINGSECTI01". Because the system segments FILESYS1A and CACHESEG had a high amount of CPU cost attributed to them, we can assume that many file system intrinsics were called. This is confirmed when we go to the Procedure and Address Level histograms.

The remainder of the case study locates and identifies the statement(s) that consumed the most direct and indirect CPU cost the statements that accrued wait times. Finally, an example shows how to translate sample counts into seconds.

5. Direct CPU Cost

Direct CPU cost represents executing statements in the program. At the Procedure and Address level of ANALYZER, histograms show what PB-locations consumed CPU cost. The VERBS map is used to correlate a specific range of PB-locations in the histograms to statements in TESTPROG.

Procedure and Address Level:

At this level of presentation detailed CPU execution profiles of the code within each segment can be obtained. Enter a profile command to get segment prompts. Parentheses indicate alternative commands.

Procedure and Address Level Commands:

DP (IP): Direct (& Indirect) CPU Time by Procedures
 DR (IR): Direct (& Indir.) by Procedure-Relative Addresses
 ^D (I): Direct (& Indirect) by Segment-Relative Addresses
 W (WR): Dir+Indir+Wait by Segment (Procedure) Rel. Addr.

LM >= n: Set Bar Threshold >= n%
 LM > n: Set Bar Threshold > n% (LM > 0% is default)

| | | |
|---------------|----------------|----------------------------|
| A: Automatic^ | M: Menu | CL: Clear Screen** |
| E: End | QU: Quiet | RL: Roll Up |
| EX: Exit | VQ: Very Quiet | HC: Hard Copy only |
| H: Help | VB: Verbose** | TD: Terminal Display only |
| T: Top | | TH: Terminal & Hard Copy** |

ANALYZER-> D

Segment prompts are displayed in the order of decreasing CPU utilization. To each prompt respond with one of the following:

YES or [RET]: To Produce Profile With Default Resolution
 <num> : Profile With Resolution Equal to <num> Words
 N or NO : Skip This Segment
 E : Terminate Prompts

User Segments:

and logically mapped SL segments:

Segment WORKINGSECTI01' (%000) (30 Samples = 7.44% Direct) ?4
 Segment MAINLINESECT00' (%001) (2 Samples = .49% Direct) ?4
 Segment COBLIB18 (%003) (6 Samples = 1.48% Direct) ?E

System Segments:

System Segment FILESYS1A (%135) (138 Samples = 34.24% Direct) ?E

**** Processing of log file begun. Please wait. ****

```

Program Name(s):TESTPROG.JORDAN.SS - Segment:WORKINGSECTI01' (%000)
Distribution of Direct CPU Time Over Segment-Relative Addresses
( 30 Segment Samples = 7.44% Direct - Including OnICS samples)
+-----+-----+-----+-----+-----+-----+-----+
%000040 000043 DD | 3.3
%000060 000063 DD | 3.3
%000064 000067 DDDD | 6.7
%000070 000073 DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD | 86.7
+-----+-----+-----+-----+-----+-----+-----+
Minimum bar threshold is .0%

Program Name(s):TESTPROG.JORDAN.SS - Segment:MAINLINESECT00' (%001)
Distribution of Direct CPU Time Over Segment-Relative Addresses
( 2 Segment Samples = .49% Direct - Including OnICS samples)
+-----+-----+-----+-----+-----+-----+-----+
%000020 000023 DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD | 100.
+-----+-----+-----+-----+-----+-----+-----+
Minimum bar threshold is .0%

ANALYZER->

```

The above histograms are for segments "WORKINGSECTI01" and "MAINLINESECT00". Those segments are the only user segments that consumed CPU time in the program TESTPROG.

The two histograms cannot be compared with each other to determine which segment consumed the most CPU time relative to the other. ANALYZER displays each histogram in decreasing CPU utilization. This is based on the number of samples collected for each segment.

In general, you must keep in mind the total number of samples described in each histogram compared to the total number of samples for the entire measurement.

When a program contains a significant portion of time spent within a small range of PB-locations, it is called a "hot spot".

To locate a hot spot in the compiler listing the following steps must be performed:

1. Review a histogram and pick an address range that you want to find in the compile listing.
2. Find in the VERBS map listing the segment name that matches the segment name in the histogram.
3. Searching down from the start of the correct segment in the VERBS map, find the PB-locations that match the hot spot in the histogram. These addresses do not always match up. The example will clarify how to match a histograms PB-locations with the VERBS map PB-locations.

4. Once the PB-locations have been found, look to the left for the compiler listing line number.
5. Now, go to the compiler listing and search down the listing until the matching line number has been found.

Example: Find the hot spot for the segment WORKINGSECTI01'.

1. The most CPU intensive hot spot occurs between PB-locations %70 and %73 in segment WORKINGSECTI01'.
2. Finding the segment WORKINGSECTI01' and PB-locations %70 in the VERBS map listing, we get:

| LINE # | PB-LOC | PROCEDURE NAME/VERB | INTERNAL NAME |
|--------|--------|---------------------|---------------|
| 00048 | 000065 | MOVE | |
| 00049 | 000076 | C100-EXIT | |

The PB-locations %70 through %73 fall between %65 and %76. So, PB-locations %70 through %73 are part of the MOVE statement. The actual number of machine instructions to accomplish the MOVE is PB-locations %65 through PB-locations %75 or %11 machine instructions.

3. Search down the compiler listing line numbers for line number 48. The statement is:

```
00048  005300  MOVE SPACES TO WRITE-LINE.
```

This makes sense because moving 2000 spaces to the array WRITE-LINE does require a fair amount of CPU time.

6. Indirect CPU Cost

Indirect CPU cost represents the execution time of called procedures residing in SLs. The histograms produced combine the Direct and Indirect CPU utilization together. The same steps are required to locate indirect hot spots as direct hot spots. This section covers how to produce indirect histograms and a walk through one example.

Procedure and Address Level:

At this level of presentation detailed CPU execution profiles of the code within each segment can be obtained. Enter a profile command to get segment prompts. Parentheses indicate alternative commands.

Procedure and Address Level Commands:

```
DP (IP): Direct (& Indirect) CPU Time by Procedures
DR (IR): Direct (& Indir.) by Procedure-Relative Addresses
^D (I): Direct (& Indirect) by Segment-Relative Addresses
W (WR): Dir+Indir+Wait by Segment (Procedure) Rel. Addr.
```

```
LM >= n: Set Bar Threshold >= n%
LM > n: Set Bar Threshold > n% (LM > 0% is default)
```

| | | |
|---------------|----------------|----------------------------|
| A: Automatic^ | M: Menu | CL: Clear Screen** |
| E: End | QU: Quiet | RL: Roll Up |
| EX: Exit | VQ: Very Quiet | HC: Hard Copy only |
| H: Help | VB: Verbose** | TD: Terminal Display only |
| T: Top | | TH: Terminal & Hard Copy** |

ANALYZER-> I

Segment prompts are displayed in the order of decreasing CPU utilization. To each prompt respond with one of the following:

```
YES or [RET]: To Produce Profile With Default Resolution
<num> : Profile With Resolution Equal to <num> Words
N or NO : Skip This Segment
E : Terminate Prompts
```

User Segments:

```
Seg. WORKINGSECTI01' (%000) [381 Samples=(7.57% D & 88.63% I)] ?4
Seg. MAINLINESECT00' (%001) [ 15 Samples=(.50% D &  3.28% I)] ?4
```

**** Processing of log file begun. Please wait. ****

```

Program Name(s): TESTPROG.JORDAN.SS - Segment: WORKINGSECTI01' (%000)
Distribution of Direct & Indir. CPU Time Over Seg Relative Addresses
( 381 Samples =  7.57% Direct &  88.63% Indirect )
+-----+-----+-----+-----+-----+-----+-----+
%000030 000033  IIIII  | 9.7
%000040 000043             |.3
%000050 000053  IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII  | 77.4
%000060 000063             |.3
%000064 000067             |.5
%000070 000073  DDDD  | 6.8
%000100 000103  III  | 4.7
%000144 000147             |.3
+-----+-----+-----+-----+-----+-----+-----+
Minimum bar threshold is  .0%

```

```

Program Name(s): TESTPROG.JORDAN.SS - Segment: MAINLINESECT00' (%001)
Distribution of Direct & Indir. CPU Time Over Seg Relative Addresses
( 15 Samples =  .50% Direct &  3.28% Indirect )
+-----+-----+-----+-----+-----+-----+-----+
%000020 000023  DDDDDDDDDDDDD  | 13.3
%000074 000077  IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII  | 40.0
%000100 000103  IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII  | 46.7
+-----+-----+-----+-----+-----+-----+-----+
Minimum bar threshold is  .0%

```

ANALYZER->

Finding the hot spot requires the following steps:

1. Determine the address range pair that consumed the most amount of CPU utilization. It is in segment WORKINGSECTI01' at PB-locations %50 through %53.
2. Search down the VERBS map listing to find the segment WORKINGSECI01'. Find the PB-location in which PB-locations %50 through %53 fall between. This is line number 46 (a READ). The READ begins at %36 and ends at %53.
3. Search down the compiler listing for line number 46. The statement is:

READ DATA-FILE AT END MOVE "Y" TO EOF.

It seems reasonable that this is a hot spot because it is performed 100 times each time through the main loop.

7. Wait Time

Wait time is the accumulated total number of SAMPLER interrupts during which the program being studied was not active. This non-active period can occur for I/O delays, user think time, and delays due to system load factors. The histograms produced combine the Direct, Indirect, and Wait Times together. This section shows how to calculate Wait Times in terms of seconds.

Procedure and Address Level:

At this level of presentation detailed CPU execution profiles of the code within each segment can be obtained. Enter a profile command to get segment prompts. Parentheses indicate alternative commands.

Procedure and Address Level Commands:

DP (IP): Direct (& Indirect) CPU Time by Procedures
 DR (IR): Direct (& Indir.) by Procedure-Relative Addresses
 ^D (I): Direct (& Indirect) by Segment-Relative Addresses
 W (WR): Dir+Indir+Wait by Segment (Procedure) Rel. Addr.

LM >= n: Set Bar Threshold >= n%
 LM > n: Set Bar Threshold > n% (LM > 0% is default)

| | | |
|---------------|----------------|----------------------------|
| A: Automatic^ | M: Menu | CL: Clear Screen** |
| E: End | QU: Quiet | RL: Roll Up |
| EX: Exit | VQ: Very Quiet | HC: Hard Copy only |
| H: Help | VB: Verbose** | TD: Terminal Display only |
| T: Top | | TH: Terminal & Hard Copy** |

ANALYZER-> W

Segment prompts are displayed in the order of decreasing Direct+Indirect+Wait times. To each prompt respond with one of the following:

YES or [RET]: To Produce Profile With Default Resolution
 <num> : Profile With Resolution Equal to <num> Words
 N or NO : Skip This Segment
 E : Terminate Prompts

Note that D stands for Direct, I for Indirect and W for Wait percentages.

User Segments:

Seg MAINLINESECT00' (%001) [3158 Intrpts=(.04% D,.27% I & 65.68% W)] ?4

Seg WORKINGSECTI01' (%000) [1634 Intrpts=(.62% D,7.33% I,26.18% W)] ?4

**** Processing of log file begun. Please wait. ****

DISC PERFORMANCE - WHAT IS IT?

RICK ALDINGER
HEWLETT-PACKARD COMPANY
11413 CHINDEN BLVD.
BOISE, IDAHO 83709

A typical HP 3000 environment measures disc performance and system throughput in transactions per hour or disc I/O's per second. Many MIS directors are interested in ways to get more out of their disc drives and CPU's. The purpose of this paper is to provide the audience with an overview of the factors involved in disc performance and some features available to you for maximizing performance.

The paper is divided into two separate segments. The first is a general discussion of the basics of disc performance as they relate solely to the disc drive. It explains the components of a disc transaction and the performance implications of each.

Segment Two discusses ways in which performance can be improved. This includes a discussion of cache and an explanation of RPS. It addresses ways your hardware configuration can affect performance as well as the role file management plays.

Any discussion of disc drives should begin with an understanding of the disc and its operation. In discussing disc performance it is necessary to first understand how a disc drive operates. Specifically, how transactions occur.

Disc Transaction

A disc transaction is comprised of four components: disc controller overhead, physical seek, physical latency, and lastly, the actual transfer of information. Transfer time is the smallest component of the transaction, with controller overhead being next. The mechanical seek and latency comprise the largest part of the transaction time.

The following chart compares the transaction components of Hewlett Packard disc drives:

| | controller | seek | latency | transfer | |
|------|------------|------|---------|----------|----|
| 7912 | 4.0 | 27.1 | 8.3 | 1.2 | ms |
| 7914 | 4.0 | 28.1 | 8.3 | 1.2 | ms |
| 7933 | 4.5 | 24.0 | 11.1 | 1.0 | ms |
| 7945 | 10.1 | 30.0 | 8.3 | 2.0 | ms |

Disc Controller

The disc controller provides the intelligence of a transaction electronically. It begins processing the transaction by:

- o Decoding the disc command sent by the host computer,
- o executing that command,
- o and finally, reporting the execution status back to the host.

The intelligence of the controller comes at the price of overhead in the disc transaction. However, experience has allowed us to make our controllers efficient in doing the greatest amount of work in the smallest amount of time.

Seek Time

Once the controller has decoded the command, the disc must perform some mechanical functions to prepare for its execution. The drive must first find the desired disc location by moving its heads to the correct media track. The mechanical movement of the head to the desired track is defined as the seek.

The time to find the desired track varies depending upon its location on the media and the current position of the head. A more accurate estimate of seek time is the AVERAGE SEEK, or the time to do all possible seeks divided by the total number of seeks possible.

Latency or Rotational Delay

Now that the drive has found the correct track it must now find the desired sector on that track. The media continues to rotate beneath the head as the track is searched. The time required for one rotation of the disc is defined as the

LATENCY time. While the media is rotating, the track is searched for the target sector. The rotation, like the seek, is mechanical.

This definition of latency is certainly a "worst case" time since the head may be considerably closer to the desired sector than one full rotation. A more accurate measure of rotational delay is the AVERAGE LATENCY. It is defined as the time to complete one half of a rotation.

Transfer

Once the head is positioned over the correct sector, it is time to transfer the data. TRANSFER is defined as the actual movement of data from the CPU to the disc (or vice versa).

HP defines AVERAGE TRANSFER as the average rate that data comes off the disc when reading an entire sector. Multiples of full sectors are always transferred in order to optimize performance. Partial sector transfers would require more bookkeeping and overhead.

Transaction Time

Each of the components of a disc transaction contribute to the total time involved in completing that transaction. The summation of the average time it takes to complete each component is a good measure of the total average time to perform a disc transaction.

The total AVERAGE TRANSACTION TIME for a particular disc product is defined as the sum of the average controller overhead for the product, plus the product's average seek time and rotational delay, plus the average time to transfer one kbyte of data to the product. The total average transaction time is specific to the product in question and, as we have defined it, does not take into account individual host system attributes.

The following figures are the various transaction times for Hewlett Packard disc drives:

| | |
|---------|--------|
| HP 7912 | 40.6ms |
| HP 7914 | 41.6ms |
| HP 7933 | 39.6ms |
| HP 7945 | 50.4ms |

Performance Metric

Hewlett-Packard uses the metric of I/O PER SECOND to measure disc performance. I/O per second is defined as the maximum number of disc transactions per second that a specific drive can perform at a transfer size of 1 kbyte. This measure is calculated by taking the inverse of the total average transaction time just described. It measures raw disc performance and does not take into account any system specifics. Actual performance will vary with system and application. I/O per second is a Hewlett Packard measurement and not an industry standard.

Let's go ahead and convert the transaction time of an HP 7912 into the measure of I/Os per second. We already learned that it takes the 7912 40.6 ms to transfer 1 kbyte of data. If we inverse our measure we can learn how many 1 kbyte data chunks can be transferred per unit of time. By converting milliseconds to seconds we have a measure of the number of kilobytes, or I/Os we can transfer in one second.

EXAMPLE: $1\text{kbyte}/40.6\text{ms} \times 1000 = 1\text{kbyte}/.0406 = 24.5 \text{ I/O per sec}$

The following figures are the I/O per second measurement for some other Hewlett Packard disc products. Again, this measure is for relative disc performance only and does not take into account system overhead. Actual performance will vary with system and application.

| | | |
|------|------|----------------|
| 7914 | 24.0 | I/O per second |
| 7933 | 25.3 | I/O per second |
| 7945 | 20.0 | I/O per second |

Well that wasn't so bad, was it? Now that we are aware of the drive's performance, let's focus on the options you have to improve disc drive efficiency. Please keep in mind the items in the following paragraphs are very system and application dependent. These are guidelines for you to use to help in performance tuning your system.

Disc Controller Cache

Disc controller cache is a method for improving performance. It is a RAM based storage area resident in the disc controller that provides high speed access to data. Frequently used data (directories, for example) are stored in the cache area, rather than on the disc media. For every cache access, a seek and latency are eliminated. The greater the "hit ratio" to

the cache, the greater the performance improvement!

Cache is currently implemented only on the HP 7933 and HP 7935 disc drives. We do plan to implement it on future high end disc products. It is supported on both the HP 3000 and HP 1000. Support for the HP 9000 is under investigation.

If we refer to our original transaction, we see that actual disc performance can be improved by decreasing the time it takes to perform even one component of the transaction. Little benefit is seen from improving transfer rates since they are only a very small component of the overall transaction. Controller efficiency has been fine tuned to a degree that shaves away most excess overhead. What about seek and latency, the largest components of the transaction? Better than reducing seek and latency times, controller cache can often times eliminate the need to perform either operation! MPE CACHE, resident in the CPU, eliminates disc controller overhead when the desired information can be accessed from the cache. The only component is the transfer of data out of CPU memory.

When is Disc controller cache better than MPE disc cache? A lightly loaded, non-cached system (less than 75% CPU utilization) will benefit from either MPE disc cache or controller cache. This type of environment is likely to be more I/O bound than CPU bound. Both caching schemes greatly reduce the I/O bottleneck. MPE disc cache will have a slight advantage over controller cache because there is no controller overhead involved when reading directly from main memory.

As the CPU load increases to a moderate level (75% to 90% utilization), the throughput of a system with MPE disc cache is impacted. The management of MPE disc cache must now compete for fewer available CPU cycles. MPE disc cache will continue to be faster than a non-cached system at this stage, but the potential for controller cache to become more effective greatly increases. Controller cache begins to provide the capability of leveling out CPU peaks.

When a system reaches the stage where CPU load is heavy (90% and above), the impact of MPE disc cache on system throughput can become negative. In extreme situations, the system may actually perform more efficiently with MPE disc cache turned off. In this environment controller cache provides a noticeable benefit, especially when MPE disc cache logical read/write ratios and read hit percentages are high. This indicates that a good deal of I/O is being eliminated and that CPU cycles and memory used for managing cache can be freed for other activities.

Controller cache will continue to significantly outperform a non-cached system until the CPU load is increased to the point where the system is so CPU bound that I/O is no longer a factor.

Rotation Position Sensing

Use of Rotation Position Sensing(RPS) is another means by which performance may be improved. RPS is a disc feature designed to minimize non-productive use of the channel while waiting for the disc to locate the area at which a transfer will begin. There is a window of time after the drive receives a command and before it finds its target sector that is generally wasted. RPS allows the channel to accept another command during that window, thus utilizing the channel better.

As might be expected, RPS provides little benefit to single drive configurations, since the channel is not the bottleneck here. But, in multiple drive, multiple process configurations, RPS can help to relieve channel contention. RPS is supported on HP 791X and HP 793X disc drives and only on the HP 3000 systems.

For instance, when a request is generated(like a read or write) that requires a disc I/O, the CPU sends a command across the channel to the disc drive. There is a window of time, after the drive receives the command and before it finds the target address on the media, that the channel is not released and cannot be used for another request. No other drives on the channel can be accessed during this window.

RPS allows the channel to be used during this window for access to the other drives on the same channel. As soon as a drive receives the command, it disconnects from the channel. During this time, other channel activity can occur. When the target address in our original transaction is found, the drive reconnects to the channel. If the channel is busy at the time, the data is buffered until the channel is free.

System Configuration

The physical location and configuration of the disc drives has a big impact on performance. Questions like, "Where do I put my system disc?", "Do each of the discs need separate interfaces?", and "When should I use multiple drives over a single, larger one?" can all be answered to optimize performance.

Please keep in mind, the optimal solution depends on the CPU in use, the number of users on the system, the application, etc. The following guidelines are conceptual and may not apply to all systems.

The location of your system disc has a big impact on performance. The optimum configuration places your system disc on a dedicated interface. With your system disc here it won't have to compete with other drives for channel activity. Since the system accesses this disc most often, a performance improvement can be realized.

As implied, channel contention can have a very negative impact on performance. Systems with many users and multiple processes accessing multiple drives can generally get relief by putting their drives on separate, dedicated interfaces. For systems with many drives this may not be economically feasible or it may exceed the maximum number of interfaces the system will permit. In these cases RPS or cache may be viable alternatives.

With the multitude of disc drive capacities to select from, it often becomes unclear which combination of drives optimizes performance. Of specific interest is the question of two smaller drives versus one larger one. The answer is very application dependent. For a system that can keep both drives busy, two drives on separate interfaces is generally the best answer. The system can then access both concurrently, increasing overall performance. On the other hand, a single faster drive is the better solution for a very localized system with little multiple processing occurring.

For instance, let's say you have a Series 68 currently configured with one IMB (inter modular bus) and two high speed GICs (general interface channels). On one GIC you have four HP 7933s, and on the other you have an HP 7978B. You would like to add an HP 7933 and an HP 2680 printer. What can you do to help increase performance?

A heavily used HP 7978 tape drive and a HP 2680 printer should have dedicated GICs. In order to do this another IMB must be added. The new IMB can then accommodate two new GICs, one each for the tape drive and printer. The HP 7933 disc drives can then be spread over the two remaining GICs on the first IMB. You could put two HP 7933 disc drives on one GIC and three HP 7933 disc drives on the remaining one.

If the tape drive is not heavily used it could share a GIC with the printer. This would free up a GIC and allow you to spread the discs out even more. This configuration would

allow your system disc to reside on a dedicated interface, and two drives on each of the remaining two GICS.

File Management

Where you physically locate your files on the discs also impacts performance. A good general rule is to spread your system files, virtual memory and other user files (including databases and spooling operations) evenly among your discs. Keeping your system files and your virtual memory on separate discs is most beneficial. By spreading your files around, the system will experience less contention in accessing the desired areas.

The biggest gain will be seen by spreading data sets and other MPE files among the various disc drives. It is also helpful in performing a reload every quarter. This helps eliminate much fragmentation that usually occurs on a heavily loaded system. When you perform the reload it is best to do an accounts reload and then restore the most heavily used files at the front of the disc. The least used files should be loaded last. This provides some benefit because the heavily used files are closer to the system directory. This helps reduce some of the disc's mechanical functions.

Overall, there are several options the performance conscious user has beyond raw disc performance. They include:

- o DISC CONTROLLER CACHE as a means of reducing mechanical seeks and latencies,
- o ROTATION POSITION SENSING to relieve channel contention,
- o SYSTEM CONFIGURATION for the most efficient use of resources, and
- o FILE MANAGEMENT to optimize throughput.

Keep in mind it may take one or all of the many items discussed to improve your performance. Each option can affect a system differently because of the applications being run and the amount of users on the system.

Do not become disenchanted if one of the options does not work. Take time to experiment with your system and the items we discussed and ask your SE and CE for help. They are an excellent resource. HP OPT/3000, HP SNAPSHOT and HP TREND are good tools in helping determine file placement, usage etc. When you take a look at your system the results may be

different each time. What might have worked one time may not produce the same high benefit the next time. However, remember that you are trying to level out the "peaks" and "valleys" on your system, providing your user with a much more balanced system.

Last but not least, use good **COMMON SENSE**.

IDENTIFYING OPPORTUNITIES FOR PERFORMANCE IMPROVEMENT

George B. Scott
ELDEC CORPORATION
16700 13th Avenue West
Lynnwood, WA 98046-0100

I. INTRODUCTION

So, the users glare at you when you walk down the hall, your phone rings with people asking if the computer is down because they've been waiting for 3 minutes for a response and your boss is telling you to fix "it". The only problem is you aren't quite sure what the "it" is. Hopefully, this paper will give you the courage to jump into the black morass of system performance problems and lead you to a situation where people greet you with smiles when you walk down the hall.

Following is a outline for the remainder of the paper.

| <u>Section</u> | <u>Title</u> |
|----------------|--|
| II | What Is Performance Optimization |
| III | Identifying the Generic Problem |
| IV | Isolating the Specific Problem & Suggested Solutions |
| V | Synergism |
| VI | Performance Monitoring |
| VII | Recommendations |
| VIII | Summary |
| A | Appendix -- Selected Tools & Reports |
| B | Appendix -- A Brief Discussion on DB Elongation |

II. WHAT IS PERFORMANCE OPTIMIZATION

Performance is not optimum whenever a process is (1) waiting on a resource or (2) using more of a resource than necessary. Thus, performance can be improved by either reducing contention for the resource or eliminating altogether the quantity of the resource being used.

The most common resources insufficient to satisfy demand are CPU availability and disc I-O capacity. To a lesser extent, the availability of memory, SIR's, RIN's or inter-CPU communication capacity/speed can be the culprit. Queuing problems usually do little to affect overall throughput; however, they may cause extreme variances in response time. To further muddy the waters, resources are commonly affected from four different origins: application code developed by programmers (or purchased from vendors), the operating system supplied by Hewlett-Packard (HP), operators who manage the system configuration and environment and, users who have an infinite variety of ways to misuse the tools available to them. Optimizing performance requires that one achieve the best overall balance among all these factors.

Having used the HP3000 since the days of the CX (1974), I have come to appreciate a certain similarity between system management and gardening. First, beyond a certain point, extra attention will do little to increase your yield. Second, if a well managed garden is left unattended, the weeds will soon take over. Hopefully, this paper will present some ideas for improving your system's performance so that you will have enough time to go home and weed your garden. To help us take care of our garden, we need some tools and the knowledge of when to use what tool. Figure 1 shows some of the commonly used tools. This list is not exhaustive and many fine alternatives are available today from a wide variety of vendors. Remember that becoming familiar with a tool and your system is more important than the particular tool you choose.

| <u>Class</u> | <u>Name</u> | <u>Performance Problem</u> | <u>Source</u> | <u>Cost</u> |
|--------------|----------------|---|---------------|-------------|
| Program | OPT | CPU, I-O, Memory, System Tables, RIN's, SIR's, etc. | HP | \$6.4K |
| Program | SAMPLER | CPU | HP | \$2K |
| Program | FREE | Disc Fragmentation | HP | Free |
| Program | DBLOADNG | Data Base Elongation | HP/CSL | Free |
| Program | PROFILER | Data Base I-O, CPU, Locks, Buffers | HP | \$3.5K |
| Program | TUNER | System Tables | HP/CSL | Free |
| Program | SURVEYOR | CPU, I-O | HP | Free |
| Program | LOGUTIL | System Logfiles | CSL | Free |
| Program | SYSLG | System Logfiles (CPU, I-O) | Swaptape | Free |
| Utility | System Logging | I-O, CPU | HP | Free |
| Command | SHOWCACHE | I-O | HP | Free |
| Command | SHOWQ | Dispatch Q, Queue Overlap | HP | Free |
| Intrinsic | TIMER | Useful in user logging for elapsed time traps. | HP | Free |
| Intrinsic | PROCTIME | Useful in user logging for CPU time traps. | HP | Free |
| Intrinsic | JOBINFO | Useful in user logging for job information. | HP | Free |
| Intrinsic | PROCTIME | Useful in user logging for process information. | HP | Free |

Figure 1 -- Performance Tools

III. IDENTIFYING THE GENERIC PROBLEM

A computer is typically doing on of the following five things:

- | | | |
|---------------------------|-------|----------------------|
| 1. Actively Using the CPU | ----- | CPU BUSY |
| 2. Waiting on I-O | ----- | PAUSED FOR DISC |
| 3. Swapping Memory | ----- | PAUSED FOR SWAP |
| 4. Waiting on a Lock | ----- | SIR's, RIN's, LOCK's |
| 5. Overhead Management | ----- | OVERHEAD |

Of these, the user can directly influence and control the first four. To determine what your system is doing, run a program such as OPT, SURVEYOR, or RADAR. If you have any significant quantity (>5%) of PAUSE FOR DISC, then you can probably improve your performance by eliminating I-O.

If you observe any significant quantity of PAUSE FOR SWAP, then you have memory pressure and need to reduce memory use. Using OPT or a similar tool, memory pressure is also indicated by the "CLOCK CYCLE RATE" in the memory management display of the CPU-memory manager context (#C,M for OPT users).

Using OPT or a similar tool, you can observe whether processes are consistently waiting on SIR's or RIN's.

If you aren't experiencing memory pressure, waiting on a SIR or RIN, or waiting on I-O, then if you want to improve performance, you must reduce CPU use.

Typically causes of each of the above classes of problems are detailed in Figure 2 and are discussed in detail in Section IV.

| <u>Problem</u> | <u>Cause(s)</u> |
|-----------------|---|
| Excessive I-O | Blocking Factor Data Base (DB) Elongation DB Master Too Full (Synonym Chains) DB Long Sorted Chains DB Serial vs. Keyed Access Disc Fragmentation (including Virtual) Sorts -- External rather than Internal Cachecontrol -- Random/Sequential |
| Excessive CPU | Excessive I-O (file reads and writes, DB gets, puts and deletes) Opens/Closes (Files, DB's, Terminals) Launches (Runs, Creating Processes) |
| Memory Pressure | Excessive Stack Space Unnecessary Open Files/DB's |
| Lock Contention | FLOCK, DBLOCK, LOCKLOCIN, LOCKGLORIN |
| Queuing | Quantum Overlap Range DS/3000 |

FIGURE 2 -- PERFORMANCE PROBLEMS / CAUSES

IV. ISOLATING THE SPECIFIC PROBLEM AND SUGGESTED SOLUTIONS

This section contains a brief explanation of common problems and some ways to identify the specific cause of the problem. The fix or solution, which is usually simple, is recommended for common problems.

I-O Problems -- Blocking Factor

Solutions to I-O problems can be made by reducing I-O. To find who is causing the I-O, turn on system logging and analyze the log files using one of the many tools available. Figure 3 shows a few lines from the contributed tool SYSLG. By sorting this file by blocks, one can readily see which files are receiving the greatest amount of I-O. If the block factor is not large, (some experts suggest 4K block size), then increase the blocking factor. This has a secondary effect which will be discussed in the section on reducing CPU.

For example, in the case shown in Figure 4, the following files are candidates for reblocking: PA520D,PA521D,PA510D,W0700D, PA530D and PDMPX. The files SORTSCR and SL cannot be reblocked because MPE requires them to have a blocking factor of 1.

| ACCOUNT | GROUP | FNAME | CLASS | LDEV | #RECS | #BLKS | #OPENS |
|---------|----------|----------|-------|------|--------|--------|--------|
| ELDEC | .DATAIR | .SORTSCR | DISC | 46 | 302143 | 302143 | 21 |
| ELDEC | .DATAIR | .PA520D | DISC | 25 | 17876 | 17876 | 12 |
| ELDEC | .DATACEN | .SORTSCR | DISC | 20 | 15369 | 15369 | 8 |
| ELDEC | .DATAMCD | .SORTSCR | DISC | 2 | 7629 | 7629 | 32 |
| ELDEC | .FUB | .SL | DISC | 14 | 5826 | 5826 | 54 |
| SYS | .DATADP | .DP002C | DISC | 38 | 65633 | 4285 | 109 |
| ELDEC | .DATAIR | .PM030D | DISC | 36 | 70210 | 4134 | 5 |
| ELDEC | .DATAIR | .PA101D | DISC | 66 | 69618 | 4098 | 3 |
| ELDEC | .DATAIR | .PA100D | DISC | 01 | 66327 | 3903 | 3 |
| ELDEC | .DATAIR | .PA521D | DISC | 36 | 7744 | 3872 | 2 |
| ELDEC | .DATAIR | .PA510D | DISC | 25 | 3250 | 3250 | 2 |
| ELDEC | .DPP | .SL | DISC | 55 | 2816 | 2816 | 54 |
| ELDEC | .DATAMCD | .W0700D | DISC | 96 | 2384 | 2384 | 3 |
| ELDEC | .DATAIR | .PA530D | DISC | 25 | 1625 | 1625 | 1 |
| ELDEC | .DATAIR | .PDMPX | DISC | 66 | 2730 | 1366 | 1 |
| ELDEC | .DATAIR | .PMPA12 | DISC | 66 | 23206 | 1366 | 2 |
| ELDEC | .DPP | .Z | DISC | 59 | 20257 | 1269 | 16 |
| ELDEC | .DATADP | .DP025IM | DISC | 0 | 32008 | 1120 | 4 |

FIGURE 3 -- FCLOSE Statistics from System Logfile Analysis
(One summary record for each unique filename)

Another report which can be generated from analysis of the system logfiles is shown in Figure 4. Each FCLOSE is shown along with the job/session which produced the I-O. Thus one can readily discern who is responsible for the I-O and fix the appropriate job or rebuild the file or educate the user.

In the previous case where the file PA520D was of interest, the report from SYSLOG which shows individual file closes shows that the only access to the file was by PA501J, PA502J and PA595IRJ. Thus in this case, the file could be reblocked and the job streams checked to determine what programs may have accessed the file.

| ACCOUNT | GROUP | FNAME | YYMMDD | HH:MM | J/S# | JOBNAME | LDEV | CLASS | #BLKS | #RECS |
|---------|---------|---------|--------|-------|-------|----------|------|-------|-------|-------|
| ELDEC | .DATAIR | .PA520D | 860609 | 22:10 | J0933 | PA501J | 3 | DISC | 1625 | 1625 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:10 | J0933 | PA501J | 3 | DISC | 1625 | 1625 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:12 | J0933 | PA501J | 3 | DISC | 0 | 0 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:12 | J0933 | PA501J | 2 | DISC | 0 | 0 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:14 | J0933 | PA501J | 2 | DISC | 1625 | 1625 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:18 | J0933 | PA501J | 2 | DISC | 1625 | 1625 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:19 | J0934 | PA502J | 2 | DISC | 2 | 2 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:22 | J0934 | PA502J | 2 | DISC | 1625 | 1625 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:40 | J0934 | PA502J | 2 | DISC | 4874 | 4874 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:41 | J0934 | PA502J | 2 | DISC | 1625 | 1625 |
| ELDEC | .DATAIR | .PA520D | 860609 | 22:56 | J0935 | PA503J | 2 | DISC | 1625 | 1625 |
| ELDEC | .DATAIR | .PA520D | 860609 | 23:02 | J0936 | PA595IRJ | 2 | DISC | 1625 | 1625 |

FIGURE 4 -- FCLOSE Statistics from System Logfile Analysis
(One record for each unique FCLOSE)

I-O Problems -- DB Elongation

Data base elongation occurs when the next record for a chain is expected to be in the current block, but is in a different block. This requires IMAGE to have to do another FREADDIR and transfer the block to the data base control block. The only fix is to repack the dataset sorted by key value. This can be accomplished by unloading the set, sorting the data by the key value and reloading the set. A variety of tools are available to assist in this effort. For small data bases, (less than 100,000 records), the entire database can usually be unloaded, sorted and reloaded within a few hours. For large data bases or sets, consider using one of the commercially available packages, or just unloading, sorting and reloading a single set. It is not uncommon to experience a 25% to 40% decrease in wall and CPU time for accessing a data base after a repacking. Data base elongation is reported using DBLOADNG.PRIV.TELESUP. A sample report from DBLOADNG is shown in Appendix A. Appendix B contains a more detailed description of DB Elongation for those newcomers to the subject.

I-O Problems -- DB Master Too Full (Synonym Chains)

Typically, if master sets become too full, the number of secondary entries increase and the number of contiguous full blocks increase. Either of these events, which can be determined using DBLOADNG, cause IMAGE to do many more FREADDIR's to find the entry for which it is searching or for an available slot to put an entry. This problem can be easily cured by expanding the capacity of the master set. In some cases, a different key value structure may be required to prevent clustering in the master set. Figure 5 shows the effect of master loading on the time for a DBPUT. Note that for small masters, the impact is trivial (so is the extra space!). However for larger masters, the effect is dramatic as the set exceeds 95% full. For master sets with capacities in the 100,000 entries or greater range, the effect of a too full master is disastrous to performance.

DBPUT TIME vs. % FILL OF MASTER SET

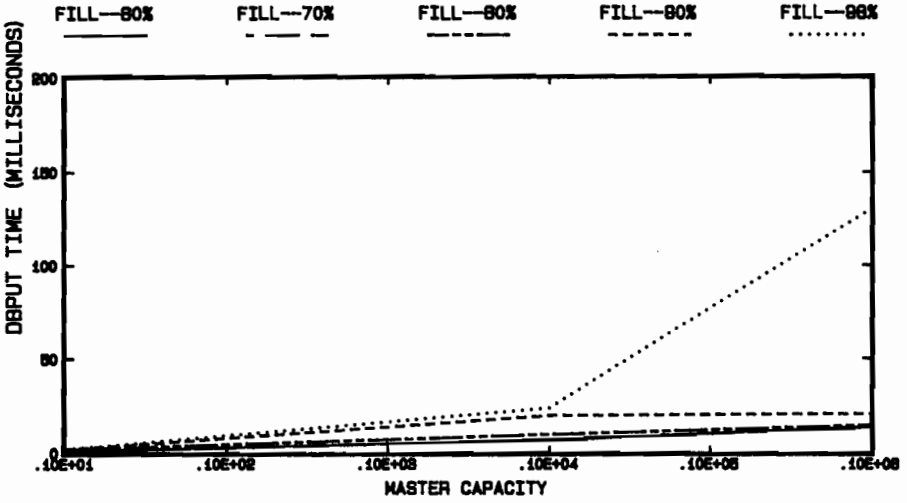


FIGURE 5 -- DBPUT Time vs. % Loading for DB Master Sets

I-O Problems -- Long Sorted Chains

Excessive I-O can be caused if you have sorted chains and a particular key value has many entries. When a DBPUT is done to a sorted chain, IMAGE starts reading records at the end of the chain and reads backward until a key value is found less than the value of the record being added or until it comes to the beginning of the chain. In the case of very long chains, the DBPUT may require several thousand disc reads prior to finding the record to which to link the new entry. This can take several seconds, or even several minutes! There is no cure for this problem; however, it is recommended that you seriously consider not using key values which might propagate long sorted chains. See Figure 6 for the effect of sorted chain length on DBPUT time.

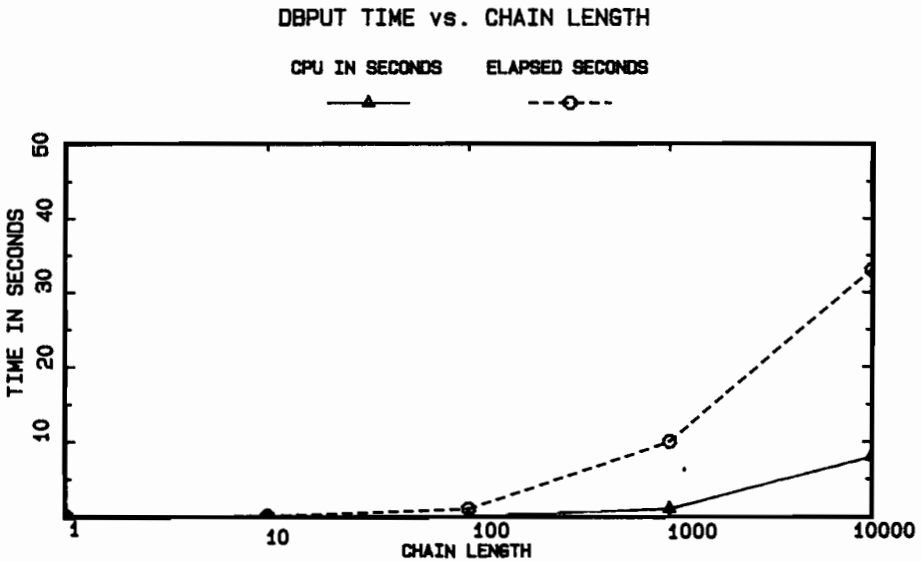


FIGURE 6 -- DBPUT Time vs. Sorted Chain Length

I-O Problems -- Serial Read vs. Key Access vs. Chain Length

Users using a utility such as QUIZ or QUERY may access data using serial reads rather than keyed reads. Rather than making a lot of snide comments about users, just identify who they are (analyze the system logfiles for I-O) and show them how they can get their reports faster. They'll love you for it.

Another trap which is easy to fall into is to not consider the frequency which short chains may be searched for a specific entry. For example, consider a parts list which has a chain on the parent part. Such a chain may be read many times looking for a specific component. If the average chain length is 20, then a average of 10 reads are required to find a specific parent-component relationship. The fix is to add a path on a concatenated field of parent-component such that the desired entry can be found in one detail read. The overhead of maintaining the extra chain is minimal compared to the cost of repeated searching of short chains, especially in situations where the DBGET to DBPUT/DBDELETE ratio is large.

As an example, last year at our site, a situation was discovered in a DB data set in which a number of our on-line transactions were reading an order number chain looking for a specific part number. A new field was added which was a concatenated order number, part number. A path was added to this field. As a result, the I-O for the affected transactions was reduced to 68% of the original, the CPU to 60% of the original and the elapsed transaction time to 57% of the original. Another phenomenon was also observed and will be discussed in the section on SYNERGISM.

I-O Problems -- Disc Fragmentation

Disc fragmentation causes excessive I-O by the operating system (MPE). Having to search for a space large enough to build the file or next extent causes extra overhead. It also increases the probability that the various extents of a file will be widely spread across the disc(s), thereby increasing the probability of disc head movement. If I-O is not a bottleneck, then this is probably not a serious problem, except when you can't build the file at all because no contiguous space on disc is big enough.

The same phenomenon occurs in virtual memory, except there is no way to cure it. Virtual fragmentation problems can be lessened by not running over a 70% use of virtual.

Disc fragmentation is shown using FREE5.PUB.SYS. The fragmentation problem can be fixed with disc condenses or system reloads. Virtual use is shown by such tools as OPT or TUNER. Figure 8 shows a sample report from FREE5.PUB.SYS. In our system, fragments less than 1000 in size are usually wasted space since our spooler is set with a 1280 minimum extent size. Thus at the time this report was run, 25% of the available space was wasted and only 295,108 sectors were actually really available for normal system use.



FREE5 G.02.A0 (C) HEWLETT-PACKARD CO., 1983

VOLUME MH7933D0 IDEV 1

LARGEST FREE AREA= 77121

| SIZE | COUNT | SPACE | AVERAGE |
|---------|-------|--------|---------|
| >100000 | 0 | 0 | 0 |
| >10000 | 2 | 108858 | 54429 |
| >1000 | 7 | 17850 | 2550 |
| >100 | 36 | 12336 | 342 |
| >10 | 133 | 4012 | 30 |
| >1 | 1314 | 3755 | 2 |

TOTAL FREE SPACE=146811

VOLUME MH7933D1 IDEV 2

LARGEST FREE AREA= 3394

| SIZE | COUNT | SPACE | AVERAGE |
|---------|-------|-------|---------|
| >100000 | 0 | 0 | 0 |
| >10000 | 0 | 0 | 0 |
| >1000 | 2 | 5803 | 2901 |
| >100 | 67 | 25361 | 378 |
| >10 | 120 | 2892 | 24 |
| >1 | 1254 | 3785 | 3 |

TOTAL FREE SPACE=37841

VOLUME MH7933D2 IDEV 3

LARGEST FREE AREA= 67634

| SIZE | COUNT | SPACE | AVERAGE |
|---------|-------|--------|---------|
| >100000 | 0 | 0 | 0 |
| >10000 | 4 | 118603 | 29650 |
| >1000 | 26 | 43994 | 1692 |
| >100 | 82 | 31005 | 378 |
| >10 | 441 | 12257 | 27 |
| >1 | 1015 | 3175 | 3 |

TOTAL FREE SPACE=209034

SYSTEM TOTAL FREE SPACE=393686

FIGURE 7 -- Disc Fragmentation as Reported by FREE5

I-O Problems -- Sorts: External vs. Internal

Still hanging over from the early days of computing are programmers and programs which sort file A to B, then read B and write a report. This technique wastes all the I-O required to write the file B and then read the file B. Do internal sorts using SORTINPUT and SORTOUTPUT, not SORT "A" GIVING "B".

I-O Problems -- CACHECONTROL, Random & Sequential

In some systems (at least in our system), the number of users varies dramatically between daytime hours (7AM-7PM) and the nighttime. In the daytime with lots of users, memory is used by the many process stacks, user data segment and file control blocks. During the evening or weekend, only a few processes are typically running at any one time. Consider increasing the cache block size during the night/weekend time to take advantage of the increase in memory availability.

EXCESSIVE CPU

Books could be written on this subject. It seems as though the more you learn about the consumption of CPU by the system, the more questions that arise; however, the following points should provide some insight into what is usually a black box in the HP3000 world.

Excessive CPU -- Excessive I-O

For those of you who thought we were through talking about I-O, let me merely say "Not yet". In many systems running typical business or manufacturing applications, the biggest single user of CPU is the I-O system.

One tool which dramatically demonstrates where CPU is used by the system is SAMPLER (APS/3000) marketed by HP. Figure 8 shows a large on-line accounting and manufacturing system and how the CPU is consumed among all system segments. Note that only 4% of the total time consumed is within the application code itself. The other 96% is within the system SL and is mostly related to the I-O system.

| <u>Percent</u> | <u>Function</u> |
|----------------|--|
| 26.0 | Morgue 'Abort, KernalC, KernalD, Miscseg, etc. |
| 12.5 | FILESYS segments |
| 29.5 | CACHESEG, HARDRES, Terminal monitors, etc |
| 10.0 | IMAGE |
| 10.0 | Communication and DS segments |
| 6.5 | V/3000 segments |
| ----- | |
| 96.5 | *SUBTOTAL |
| 3.5 | User Segments |

FIGURE 8 -- Distribution of CPU Utilization, All Segments

Another report from SAMPLER shows the segment relative addresses for the distribution of Indirect CPU utilization by a segment. By looking at a compile listing for the listed addresses, one can determine what routines are indirectly consuming the CPU. Thus if you look at how the main driver in our system, which accounts for 53% of all indirect CPU usage, indirectly consumes CPU, you will find that it is mostly spent within the I-O system as shown in Figure 9.

| Percent | Function |
|---------|--|
| 37.2 | Displays screen, reads it, reads a DB record |
| 15.2 | Reads a terminal screen |
| 6.2 | Displays messages to user screen |
| 5.5 | Gets field number from V/3000 |
| 10.8 | Logs a transaction |
| 6.4 | Opens a DB |
| 2.9 | Opens a message catalog |
| 2.3 | Opens a file for diagnostic messages |
| 1.7 | Opens 2 logfiles |
| 1.1 | Opens 3 IPC files |
| 3.4 | Opens a V/3000 formfile |
| 1.2 | Reads an EDS |
| 0.2 | Activates father process |
| 94.1 | *SUBTOTAL |
| 6.9 | Miscellaneous routines |

FIGURE 9 -- Distribution of Indirect CPU over Program Segments

Another view of how the CPU spends it's time can be obtained by looking at the CPU time required for various intrinsics. Shown in Figures 10, 11 and 12 are CPU times and elapsed times in milliseconds for commonly executed intrinsics. For those especially interested, Figure 11 shows commonly executed DB intrinsics with various combinations of logging and caching. Shown in Figure 12 are a comparison of three different CPU's using the same bench mark program and data for intrinsic execution time measurements.

From these figures, it can readily be seen that I-O plays a major role in CPU utilization.

To improve performance, you must determine where the leverage is and be realistic about how much improvement can be achieved. Using our example of a system which spends 90% to 95% of its time in the system SL, if a 20% improvement were made in the logic of a program, only a 2% improvement in system throughput would be achieved; however, if a 20% reduction in I-O were achieved, one might realize an 18% improvement in throughput. In short, put your energy where the action is.

| ELAPSED | | CPU | | Intrinsic | Comment |
|---------|------|-----|------|---------------|-------------------------|
| 70 | 42XP | 70 | 42XP | | |
| 106 | 152 | 17 | 36 | FOPEN | old,ascii, for update |
| 21 | 17 | 4 | 7 | FREAD | first record |
| 1 | 2 | 1 | 2 | " | same blk |
| 18 | 10 | 2 | 3 | " | different blk |
| 48 | 62 | 3 | 7 | FREADDIR | different blk & buffer |
| 1 | 2 | 1 | 2 | FLOCK | |
| 21 | 17 | 3 | 8 | FUPDATE | |
| 31 | 28 | 2 | 3 | FGETINFO | |
| 1 | 2 | 1 | 2 | FUNLOCK | |
| 20 | 17 | 1 | 2 | FPOINT | |
| 389 | 587 | 34 | 72 | FRENAME | |
| 48 | 54 | 7 | 19 | FCLOSE | |
| 489 | 1151 | 105 | 54 | VOPEXTERM | |
| 679 | 654 | 60 | 47 | VOPEXFORMFILE | file with 20 forms |
| 90 | 74 | 6 | 14 | VGETNEXTFORM | |
| 18 | 58 | 17 | 51 | VSHOWFORM | 23 lines, 1400 char |
| na | na | 4 | 10 | VREADFIELDS | Elapsed depends on User |
| 0 | 1 | 0 | 0 | VERRMSG | |
| 38 | 65 | 11 | 24 | VCLOSEFORMF | |
| 453 | 594 | 91 | 11 | VCLOSETERM | |
| 283 | 424 | 44 | 93 | CREATEPROCESS | |
| 1 | 1 | 1 | 1 | ACTIVATE | |
| 2 | 7 | 2 | 3 | SENDMAIL | 80 char |
| 0 | 1 | 0 | 1 | MAIL | |
| 0 | 1 | 1 | 1 | RECEIVEMAIL | |
| 1 | 3 | 1 | 3 | FWRITE | to IPC file |
| 1 | 2 | 1 | 2 | FREAD | from IPC file |
| x | x | 2 | 12 | GETDSEG | 1600 WORDS |
| 1 | 1 | 1 | 2 | DMOVOUT | 40 words |
| 1 | 1 | 0 | 1 | DMOVIN | 40 words |
| 0 | 1 | 0 | 1 | GETLOCRI | one rin |
| 0 | 1 | 1 | 1 | LOCKLOCRI | |
| 0 | 2 | 1 | 2 | PUTJCV | |
| 1 | 1 | 1 | 1 | FINDJCV | |
| 1 | 0 | 0 | 0 | PROCTIME | |
| 1 | 0 | 0 | 0 | TIMER | |

FIGURE 10 -- Commonly Executed Intrinsic (Milliseconds)

| ELAPSED TIME | | | CPU TIME | | | Intrinsic | Comment |
|----------------|------|------|----------------|-----|------|-----------|-------------------------|
| LOGGING STATUS | | | LOGGING STATUS | | | | |
| NONE | ILR | RECV | NONE | ILR | RECV | | |
| 1475 | 1905 | 1627 | 112 | 164 | 122 | DBOPEN | mode 1, first user |
| 783 | 808 | 833 | 88 | 98 | 94 | " | mode 1, second user |
| 129 | 124 | 114 | 21 | 22 | 22 | DBFIND | first on detail |
| 21 | 20 | 20 | 4 | 3 | 4 | " | second on detail |
| 161 | 159 | 176 | 19 | 22 | 23 | DBGET | chain, first read |
| 2 | 2 | 2 | 2 | 2 | 2 | " | chain, same block |
| 16 | 15 | 15 | 2 | 3 | 4 | " | chain, different blk |
| 33 | 33 | 33 | 4 | 4 | 4 | " | directed different blk |
| 1 | 1 | 0 | 1 | 1 | 0 | DBLOCK | DB unconditional |
| 175 | 176 | 175 | 31 | 35 | 32 | DEUPDATE | current record |
| 545 | 672 | 567 | 88 | 109 | 92 | DBDELETE | current record |
| 144 | 221 | 144 | 20 | 32 | 21 | DBPUT | current record, 2 paths |
| 15 | 16 | 16 | 14 | 13 | 14 | DBEXPLAIN | |
| 3 | 3 | 3 | 4 | 3 | 3 | DBERROR | |
| 1 | 1 | 1 | 1 | 1 | 1 | DBINFO | mode 202, set info |
| 0 | 2 | 1 | 1 | 2 | 2 | DBBEGIN | |
| 0 | 1 | 2 | 0 | 1 | 2 | DEMEMO | |
| 1 | 36 | 64 | 1 | 4 | 4 | DBEND | |
| 0 | 1 | 1 | 0 | 1 | 1 | DBUNLOCK | |
| 90 | 112 | 112 | 11 | 18 | 18 | DBCLOSE | not last user of DB |
| 556 | 668 | 626 | 84 | 104 | 91 | " | last user of DB |

FIGURE 11 -- Wall & CPU Times (milliseconds) for DB Intrinsic
(Caching is turned off in all cases)

| * ELAPSED * | | *** CPU *** | | | Intrinsic | Comment |
|-------------|------|-------------|-----|------|-----------|-------------------------|
| 70 | 42XP | 70 | 68 | 42XP | | |
| 1475 | 1833 | 112 | 205 | 278 | DBOPEN | mode 1, first user |
| 783 | 963 | 88 | 131 | 195 | " | mode 1, second user |
| 129 | 159 | 21 | 34 | 50 | DBFIND | first on detail |
| 21 | 19 | 4 | x | 12 | " | second on detail |
| 161 | 175 | 19 | 36 | 46 | DBGET | chain, first read |
| 6 | 5 | 2 | 3 | 5 | " | chain, same block |
| 16 | 62 | 2 | x | 11 | " | chain, different blk |
| 33 | 31 | 4 | 8 | 12 | " | directed different blk |
| 1 | 3 | 1 | 1 | 2 | DBLOCK | DB unconditional |
| 175 | 304 | 31 | 43 | 70 | DBUPDATE | current record |
| 545 | 787 | 88 | 140 | 205 | DBDELETE | current record |
| 144 | 232 | 20 | 32 | 50 | DBPUT | current record, 2 paths |
| 15 | 40 | 14 | 12 | 38 | DBEXPLAIN | |
| 3 | 7 | 4 | 4 | 7 | DBERROR | |
| 1 | 7 | 1 | 2 | 3 | DBINFO | mode 202, set info |
| 0 | 1 | 1 | 0 | 1 | DBBEGIN | |
| 0 | 1 | 0 | 0 | 0 | DBMEMO | |
| 0 | 2 | 1 | 0 | 1 | DBEND | |
| 1 | 6 | 0 | 1 | 2 | DBUNLOCK | |
| 90 | 97 | 11 | 16 | 29 | DBCLOSE | not last user of DB |
| 556 | 792 | 84 | 119 | 187 | " | last user of DB |

FIGURE 12 -- CPU & Wall Times (milliseconds) for DB Intrinsic

Memory Pressure -- Excessive Stack Space

Although it is more rare on larger HP systems, many of the smaller HP systems experience memory pressure. Users can do little to influence the use of memory. Programmers, however, can have a dramatic effect. Take for example the case of the programmer who once had a program abort because of insufficient stack space. Since that time, the programmer has always prepped programs with a maxdata of 30000 (even if only 2000 was needed). Multiply 30000 by 33 different programs and you quickly get to one megabyte of essentially wasted memory. Of course, this same programmer opens all files at the beginning of his program and closes all files at the exit from the program. The fact that the program has no use for some of these files after the initial part of the program doesn't mean the system can forget about them. Instead, it may swap some things to virtual so that at the end of the program when the files are closed, they have to be swapped in again so the file can be closed. Considering that IMAGE data bases use a variety of global, buffer and lock control blocks and certainly consume a large portion of memory, the same concept is true for programs which open data bases at the beginning of the programs and don't close them when access is no longer needed, but wait until the end of the program. In short, if you aren't going to use it anymore, close it immediately.

LOCK CONTENTION

Perhaps the most difficult of all types of performance problems to identify and isolate are those involving contention for a locked resource. A common problem of this type occurs when multiple processes request DB locks. Even using item locking, large integrated systems may occasionally attempt to lock on the same data set on different items, which causes IMAGE to treat the item lock as a data set lock.

Tools such as DBUTIL reveal locks which are held at the point the display is generated, but do not give any indication of how long the locks are being held. Opt also will show locks as a SIR wait. Neither of these is particularly helpful in understanding whether a real problem exists with your locking strategy.

Four alternatives exist. You can write a trap for the DBLOCK intrinsic and log all locks, where they came from and how long they were held (trap also DBUNLOCK, DBCLOSE). This isn't going to help your performance while you are monitoring the situation, but it will give you some good insight into whether you have a problem.

The second alternative is a report from SAMPLER which shows wait times in segments. If lock delays are being experienced, then you can determine to what extent and even backtrack to the specific lines of code. This doesn't give you any insight to particular data values being locked. In many cases, the lock problem occurs because

the process is reading a particularly long chain and the value of the data would give you some insight to this problem.

The third alternative is the tool PROFILER from HP. This has a display which shows statistics on locks and also provides a formatted display which can be used to get a very clear picture of locking activity.

Finally, the first recognition that a problem exists often comes from a user who is complaining that the transaction that normally takes a few seconds now takes a minute on occasion. If you are doing some type of user transaction logging, you can trace back and see what other transactions were being processed concurrent to the lock problem. This is a less direct approach, but usually will lead you to the conflict causing the problem. Now all you have to do is develop another strategy which will eliminate the conflict. Obviously, your original strategy hasn't worked.

QUEUING PROBLEMS

Again, we come to a subject on which an entire paper could be presented. Following is a very simplistic view of queuing. Hopefully it will at least spark some thought as to whether queuing could be a problem at your site.

Typically, most shops use the CS queue for on line processing and the DS and ES queues for batch or low priority processing. Queuing is set by the TUNE command which controls four factors: the minimum clock cycle, the range of the queue, the overlap between queues and the time quantum for a queue. Range and overlap will be discussed here.

Queuing Problems -- Range

The range of a queue controls how much shorter transactions are favored compared to longer transactions. Range is specified by the "BASEPRI" and "LIMITPRI" parameters of the TUNE command. When the difference between the two levels is very small, then all processes in the queue will essentially be in a single line in which after they use their quantum, they go to the back of the line and wait their turn again. If the range is large, e.g, 152-220, then shorter transactions will receive attention more readily than long transactions. Consider what would happen in a hamburger palace at noon time if orders were taken based on the size of the order and you have a baseball team to feed. You place your order for 20 burgers and get your first burger. Then, every customer who has an order for less than two burgers will be served before you. Finally, you get your second burger. Now every customer who has an order for less than three burgers will be served before you. You can quickly see that your baseball team would never go out to lunch with you again.

Queuing Problems -- Overlap

Overlap controls how much time will be consumed by the lower queue in relationship to the higher queue. Note that the CS queue and the DS queue are not equivalent in all respects. In fact, if identical processes are run in the CS queue and the DS queue and the MAXPRI and MINIMUMPRI are set identical for both queues, the DS queue process can consume twice as much of the CPU as the CS process. This is because of the "Average Short Transaction" time which is internally calculated by MPE for the CS queue.

If on the other hand, you totally separate the queues, then whenever any process in the higher queue wants the CPU, the lower queue process is interrupted and gets no attention until the higher queue process is completed. Separated queues is an extreme situation which should not be implemented except under the most extreme conditions. Note also, that MPE has historically had problems with lockouts in situations such as this (The lower queue owns a resource which the higher queue needs, but the process in the lower queue

cannot be launched by the DISPATCHER for further process because a higher priority process is scheduled and ready.

A variety of papers and articles have been written on queuing. You should peruse these and experiment with queuing at your site to find what settings seem to give the best response to all needs.

Another phenomenon that can greatly influence performance because of queuing problems is using DS/3000 for inter-CPU communication. DSMON runs in the top of the BS queue because it must respond to requests from the INP board within a specified time limit. Now for the problem.

Consider the case where a very lightly loaded system "L" is linked to a very heavily loaded system "H". If "L" requests a transfer from "H", then "H" responds immediately (after all it is running at a priority of 100). Of course, system "L" can immediately request more data since it has no contention. Thus you have the situation where the requestor is totally dominating the server, even though the server has many processes which need attention. This situation can easily lead to system "L" consuming as much as 80% of the available CPU on the remote system. Obviously, users on an already slow system are thrilled when the response stops altogether.

For those of you lucky enough to live with this condition there is very little you can do about it other than discourage large amounts of DS traffic during peak load periods. Of course, you could have the requesting process pause with some degree of frequency; however, you have then created a situation in which the DS throughput has been drastically lowered, even if it is not necessary.

V. SYNERGISM

One of the more delightful events which can happen in performance enhancement activities is to achieve a improvement much greater than anticipated. During the discussion on repeated reading of DB chains looking for a specific item, a case was describe in which a new item and path were added to eliminate I-O and improve response times for those transactions looking for that particular data. When this was implemented, all the non-modified transactions has a net improvement in execution time reducing them to 85% of their original. Consider for example a supermarket with a cash only checkout line for people with a limited number of items. By eliminating the large time consumers from the queue, the remaining customers are serviced more quickly.

VI. PERFORMANCE MONITORING

Two things occur when you monitor performance on a routine and constant basis. First, you separate myth from fact. Users who say the response time is a half a minute are less likely to be so vocal when the precise 7 second response time was logged and you can demonstrate with a stopwatch that the logged time is correct. Furthermore, you can get an early warning from the log of degradation which might be unapparent to the user.

The second product of routine performance monitoring is learning how your system normally behaves. Thus if you make a change to the system such as adding new hardware, a new operating system, additional users or new software subsystems, you have a point of reference based on fact to determine in a quantitative manner how the change has affected your system.

Several tools exist for global performance monitoring such as OPT, RADAR, SURVEYOR, etc. Shown in APPENDIX A are several reports of interest which can be generated from these tools, either directly or indirectly.

Another approach which should be considered is user logging, especially if your system is a transaction oriented system. Shown in APPENDIX A is a report which is generated from our on-line transaction log. This summary report shows information about CPU's, divisions, transaction command-id's, users and terminals. Analysis of this report, which is generated daily, helps us determine where the greatest leverage is in improving system throughput.

A third approach to determine how seriously your system is degrading during periods of heavy loads is to create a utility which simply launches a very short transaction every 5 or 10 minutes. Run this process continuously. By observing the elapsed time for the transaction during peak load conditions, you can get a feel to the total demand being placed on your system and the variance between peak load--no load conditions.

VII. RECOMMENDATIONS

1. Start some type of performance monitoring now. Note that for system wide monitoring, it might be wise to create a group in the SYS account with system manager capability.
2. Become familiar with the tool you choose. Use it often. Take classes if they are available. Read the manual. Use it until you understand (or at least think you do) the displays/output. At that point, you should be able to correlate information from the tool with response time and throughput.
3. If at all possible, build some type of transaction logging function. If you can't do that, consider some type of IMAGE logging.
4. Turn on system logging. Get a system logfile analysis tool and find out where the I-O are going and who is consuming the CPU. Note that most all the system logfile tools are free.
5. Establish some type of performance monitoring which runs continuously. Analyze the output from this and determine where your leverage is for performance improvement. Tackle problems that represent more than five percent of the total load.
6. Become familiar with the tool DBLOADNG and consider repacking sets or data bases when they have significant elongation.
7. Avoid long sorted chains in IMAGE.
8. Provide a direct path for access to DB information with a high access frequency.
9. Go to your local user group meetings. Read Interact, Supergroup, The Chronicle and other articles.
10. Submit a paper so we can all profit from your experiences.

VIII. SUMMARY

Performance on the HP3000 is a function of load, capacity and critical resources. To better understand how performance degrades as load is added, a benchmark was run on a series 42XP in a standalone condition. The benchmark consisted of a father process which created a number of son processes. Each son process did ten I-O and then paused for one second. By varying the number of processes, the rate of degradation of performance was observed. Figure 13 shows response time vs. number of processes. Note that response time was defined as the time required to post a single I-O. Each posted record contained the CPU and elapsed time of the previous I-O.

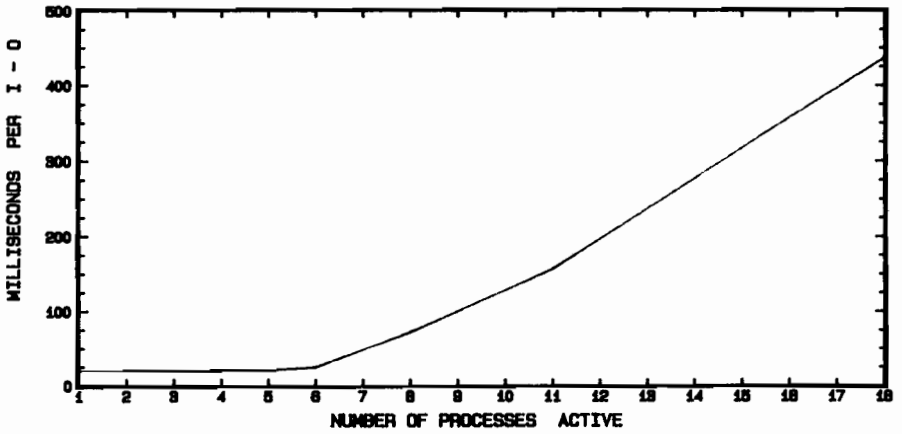
In a similar fashion, the benchmark was repeated with a process tree in which the son consumed .9CPU, posted an I-O of the transaction time and then paused for one second.

The important observation is not the particular data, but the shape of the curve as load increases and response decreases. By building a standard transaction for your system and running a similar test, you could then determine how overloaded your system is under peak load conditions by simply running the standard and determining the percent degradation.

Now that I've finished this paper, I can go home and watch my garden grow.

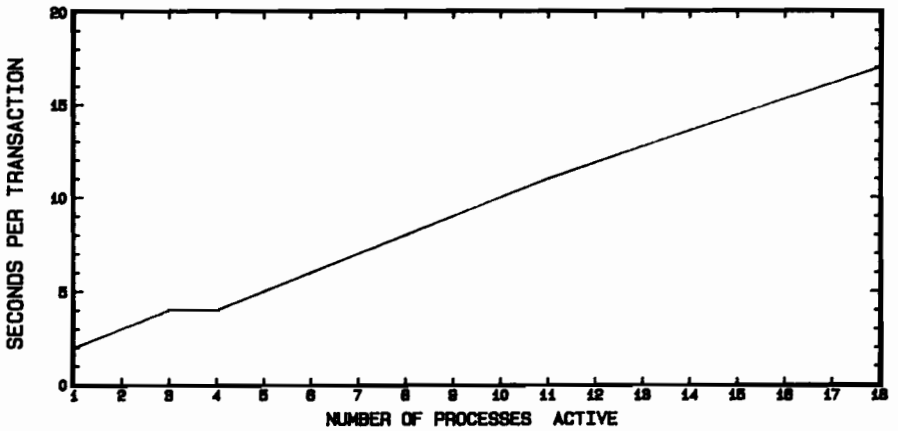
LOAD vs RESPONSE TIME for I-O INTENSIVE LOAD

MILLISECONDS/I-O



LOAD vs RESPONSE TIME for CPU INTENSIVE LOAD

SECONDS / TRANS



APPENDIX A -- SAMPLE REPORTS FROM VARIOUS PERFORMANCE TOOLS

- Figure A1 -- Report from DBLOADNG
- Figure A2 -- Report from OPT/3000
- Figure A3 -- Report derived from OPT/3000 data
- Figure A4 -- Report from SAMPLER, (Direct CPU all Segments)
- Figure A5 -- Report from SAMPLER, (CPU by Segment Rel. Addr.)
- Figure A6 -- Report from TUNER
- Figure A7 -- Report from SURVEYOR
- Figure A8 -- Report from PROFILER (DB locks)
- Figure A9 -- Report from SYSIG10 (Job Info - Proprietary)
- Figure A10-- User Transaction Log Analysis (Proprietary)

Data Base Loading Analysis for IMAGE/3000 DBLOADNG V2.3
 Data Base: DBTBL
 Run on: THU, JUN 12, 1986, 3:34 AM
 Page: 1

| Data Set | Type | Capacity | Entries | Load Factor | Sec- daries | Max Blk Blks | Fact | Search Item | Max Chain | Avg Chain | Std Dev | Excd Blcks | Avg Blcks | Ineff Ptrs | Elong- ation |
|-----------------|------|----------|---------|-------------|----------------|-----------------|------|---------------|--------------|--------------|------------|---------------|--------------|---------------|-----------------|
| TBL-ORG-MSTR | Man | 101 | 68 | 67.2% | 18.8% | 0 | 55 | TBL-ORG | 3 | 1.2 | .5 | 1.0 | 1.0 | .0% | 1.00 |
| TBL-MSTR | Atc | 7507 | 5658 | 75.4% | 29.4% | 1 | 51 | TBL-CODE-ORG | 5 | 1.4 | .7 | 1.0 | 1.0 | .8% | 1.01 |
| OR-NUM-MSTR | Man | 120011 | 75511 | 62.9% | 28.1% | 1 | 89 | OR-NUM | 7 | 1.4 | .7 | 1.0 | 1.0 | .1% | 1.00 |
| P-E-CODE-MSTR | Man | 4001 | 1858 | 46.4% | 21.2% | 1 | 33 | P-E-CODE | 4 | 1.3 | .5 | 1.0 | 1.0 | .0% | 1.00 |
| P-E-RULE-MSTR | Atc | 151 | 101 | 66.9% | 18.8% | 0 | 53 | P-E-COL-CODE | 3 | 1.2 | .5 | 1.0 | 1.0 | .0% | 1.00 |
| TERMS-CODE-MSTR | Atc | 251 | 79 | 31.5% | 6.3% | 0 | 92 | TERMS-CODE | 2 | 1.1 | .3 | 1.0 | 1.0 | .0% | 1.00 |
| DOC-DIST-MSTR | Man | 101 | 1 | 1.0% | .0% | 0 | 101 | DOC-DIST-CODE | 1 | 1.0 | .0 | 1.0 | 1.0 | .0% | 1.00 |
| DOC-STATUS-MSTR | Man | 23 | 10 | 43.5% | 30.0% | 0 | 23 | DOC-STATUS | 2 | 1.4 | .5 | 1.0 | 1.0 | .0% | 1.00 |
| PLAN-RATES-MSTR | Man | 101 | 16 | 15.8% | .0% | 0 | 51 | ORG-NUM | 1 | 1.0 | .0 | 1.0 | 1.0 | .0% | 1.00 |
| ORG-MSTR | Man | 499 | 5 | 1.0% | .0% | 0 | 84 | ORG-ID | 1 | 1.0 | .0 | 1.0 | 1.0 | .0% | 1.00 |
| TBL-DATA | Dtl | 7502 | 5928 | 79.0% | | | 11 | TBL-ORG | 3275 | 123.5 | 474.9 | 12.2 | 16.8 | 12.4% | 1.38 |
| LOCK-DATA | Dtl | 1 | 0 | .0% | | | 1 | ITBL-CODE-ORG | 17 | 1.0 | .5 | 1.0 | 1.0 | 8.9% | 1.00 |
| P-E-TITLE-RULES | Dtl | 264 | 101 | 38.3% | | | 44 | IP-E-COL-CODE | 1 | 1.0 | .0 | 1.0 | 1.0 | .0% | 1.00 |
| TERMS-DATA | Dtl | 210 | 79 | 37.6% | | | 35 | ITERMS-CODE | 1 | 1.0 | .0 | 1.0 | 1.0 | .0% | 1.00 |
| PLANNER-DATA | Dtl | 576 | 35 | 6.1% | | | 72 | SIORG-ID | 13 | 7.0 | 5.7 | 1.0 | 1.0 | .0% | 1.00 |

Elapsed Time: 169.600 Seconds
 CPU Time: 120.147 Seconds

REPORT 1

WED, JUN 11, 1986, 6:25 PM
SUMMARY REPORT EXAMPLE FROM OPT.PUB.SYS

SUMMARY REPORT HP3238A.00.26 OPT/3000
(C) HEWLETT-PACKARD COMPANY 1979, 1980
INTERVAL LENGTH: 60.608 SECONDS (1.0 MINUTES)

| CPU ACTIVITY SUMMARY | | | |
|----------------------|------|-----|------------|
| CPU STATE | MEAN | MAX | TOTAL TIME |
| CPU BUSY | 22% | 50% | 13.038 |
| PAUSE DISC & SWAP | 0% | 0% | 0 |
| PAUSE DISC | 10% | 12% | 6.117 |
| PAUSE SWAP | 0% | 0% | 0 |
| PAUSE IDLE | 6% | 7% | 3.394 |
| GARBAGE COLLECTION | 0% | 0% | 0 |
| MEMORY ALLOCATION | 22% | 25% | 13.335 |
| ICS/CACHE OVERHEAD | 40% | | 24.724 |

| MEMORY ALLOCATION SUMMARY | | | |
|---------------------------|------|-------|--|
| RESULT | MEAN | COUNT | |
| RECOVERY | 0% | 0 | |
| FREE SPACE | 71% | 630 | |
| OVERLAY CAND | 28% | 255 | |
| GIVE UP | 1% | 11 | |
| HARD REQUEST | 0% | 0 | |

LAUNCH ACTIVITY AND ADDITIONAL MEMORY MANAGEMENT ACTIVITY SUMMARY

| PROCESS LAUNCHES | PROCESS SWAP-INS | PROCESS PREEMPTS | MEMORY ALLOCS | SPECIAL REQUESTS | MM I/O READS | MM I/O WRITES | RELEASE DATA SEG | RELEASE CODE SEG | CLOCK CYCLES |
|------------------|------------------|------------------|---------------|------------------|--------------|---------------|------------------|------------------|--------------|
| 635 | 885 | 54 | 885 | 0 | 1 | 0 | 0 | 4 | 1 |
| 10.5 | 14.6 | .9 | 14.6 | .0 | .0 | .0 | .0 | .1 | .0 |
| MAX RATE | 15 | 2 | 15 | 0 | 6 | 0 | 0 | 0 | 0 |

SUMMARY OF DISC ACTIVITY

| | ALL I/O | | COUNT/RATE | | CONTROL OPS | | MAXIMUM RATE(USER/MM AND CACHE) | |
|-----------------|------------|----------|------------|--------|-------------|--------|---------------------------------|--------|
| | READS | WRITES | READS | WRITES | READS | WRITES | READS | WRITES |
| ALL DISC | 1276/ 21.1 | 235/ 3.9 | 1035/ 17.1 | 6/ .1 | 0/11 | 0/23 | 0/11 | 0/23 |
| DISC 1 (LDEV 1) | 155/ 2.6 | 29/ .5 | 126/ 2.1 | 0/ .0 | 0/ 1 | 0/11 | 0/ 1 | 0/11 |
| DISC 2 (LDEV 2) | 198/ 3.3 | 102/ 1.7 | 95/ 1.6 | 1/ .0 | 0/ 8 | 0/ 2 | 0/ 2 | 0/ 2 |
| DISC 3 (LDEV 3) | 923/ 15.2 | 104/ 1.7 | 814/ 13.4 | 5/ .1 | 0/ 2 | 0/14 | 0/ 2 | 0/14 |

SUMMARY OF LP ACTIVITY

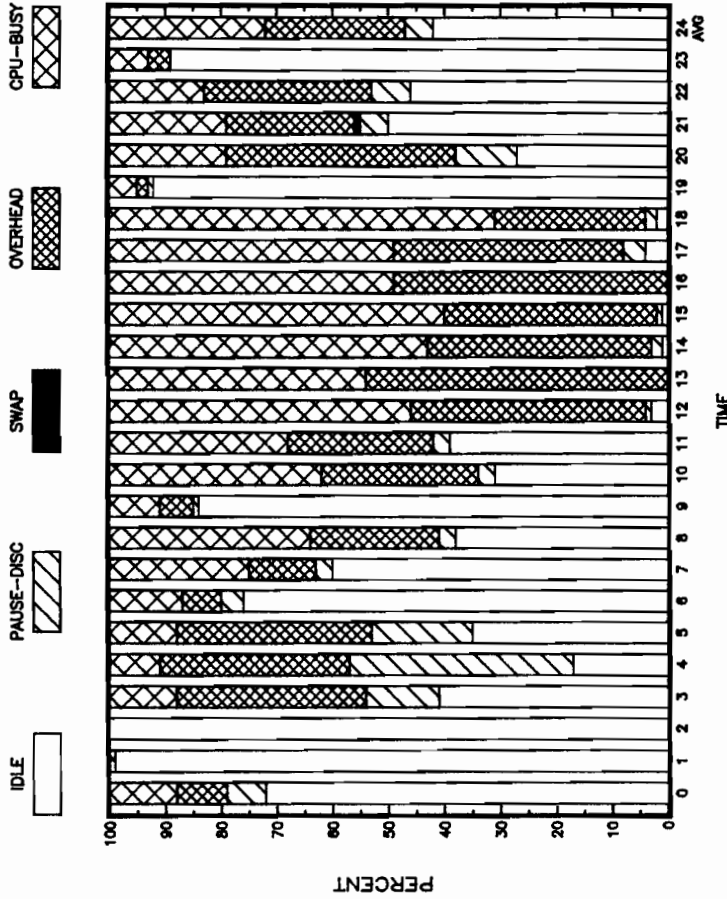
| | ALL I/O | | COUNT/RATE | | CONTROL OPS | | MAXIMUM RATE | |
|---------------|---------|--------|------------|--------|-------------|--------|--------------|--------|
| | READS | WRITES | READS | WRITES | READS | WRITES | READS | WRITES |
| ALL LP | 11/ .2 | 11/ .2 | 11/ .2 | 11/ .2 | 0/ .0 | 0/ .0 | 0/ .0 | 0/ .0 |
| LP 2 (LDEV 6) | 11/ .2 | 11/ .2 | 11/ .2 | 11/ .2 | 0/ .0 | 0/ .0 | 0/ .0 | 0/ .0 |

SUMMARY OF TAPE ACTIVITY

| | ALL I/O | | COUNT/RATE | | CONTROL OPS | | MAXIMUM RATE | |
|----------|---------|--------|------------|--------|-------------|--------|--------------|--------|
| | READS | WRITES | READS | WRITES | READS | WRITES | READS | WRITES |
| ALL TAPE | 0/ .0 | 0/ .0 | 0/ .0 | 0/ .0 | 0/ .0 | 0/ .0 | 0/ .0 | 0/ .0 |

HP3000 SYSTEM USAGE

06/09/86 --- SYSTEM D



Program SAMPLER/3000 (ANALYZER) TODAY: WED, JUN 11, 1986, 8:05 PM REPORT #2

Measurement Title: UT030P 9:00AM--9:59AM JUNE 6, 1986 SYSTEM B, SERIES 68

Sub-Title: EXAMPLE OF REPORT FROM SAMPLER

Measurement Date: FRI, JUN 6, 1986, 9:06 AM

Program Name(s): UT030P.DPP.ELDEC

Distribution of Direct CPU Utilization Over All Segments
(77462 Samples - Including OnICS samples)

| Program Name(s) | CNT | % | %CUM |
|-------------------------------|-------|------|------|
| MORQUE\AR0BT1 | 1048 | 1.4 | 2.1 |
| KERNELC1 | 8884 | 11.5 | 15.2 |
| KERNELD1 | 1010 | 1.3 | 16.5 |
| MISCSEGC\CHECK1 | 7331 | 9.5 | 26.0 |
| FILESYS\IA1 | 6750 | 8.7 | 34.7 |
| HARDRES1 | 8602 | 11.1 | 49.6 |
| TERMONITOR1 | 1568 | 2.0 | 51.7 |
| TERMANAGER1 | 1042 | 1.3 | 53.1 |
| TERMDRIVER1 | 1124 | 1.5 | 54.6 |
| CACHESEG1 | 10397 | 13.4 | 68.1 |
| TIMAGE01 | 5804 | 7.5 | 75.0 |
| TIMAGE02 | 848 | 1.1 | 74.1 |
| TIMAGE04 | 2063 | 2.7 | 76.8 |
| DSTOM1 | 4663 | 6.0 | 84.9 |
| H10MDSG2 | 1434 | 1.9 | 86.7 |
| I01NPO1 | 1368 | 1.8 | 88.7 |
| I00SX1 | 800 | 1.0 | 89.8 |
| SEG1 | 1016 | 1.3 | 93.4 |
| V3000\4 | 839 | 1.1 | 95.0 |
| V3000\6 | 1542 | 2.0 | 97.0 |
| V3000\7 | 2021 | 2.6 | 99.6 |
| Minimum bar threshold is 1.0% | CNT | % | %CUM |

(C) HEWLETT-PACKARD 32180A.01.03 Application
Program SAMPLER/3000 (ANALYZER) TODAY: WED, JUN 11, 1986, 8:05 PM REPORT #3

Measurement Title: UT030P 9:00AM--9:59AM JUNE 6, 1986 SYSTEM B, SERIES 68

Sub-Title: EXAMPLE OF REPORT FROM SAMPLER

Measurement Date: FRI, JUN 6, 1986, 9:06 AM

| Program Name(s) | CNT | % | %CUM |
|-------------------------------|-------|------|------|
| ALL SYSTEM | 77137 | 99.6 | 99.6 |
| Minimum bar threshold is 1.0% | CNT | % | %CUM |

| TABLE | CONFIG. VALUE | MAX | IN USE | CUR. | IN USE | MAX | CUR | % | MAX | % | CUR | % | SIZE/MS |
|----------|---------------|-------|--------|-------|--------|---------|-----|---|-----|---|-----|---|---------|
| DST | * 1024 | 771 | 630 | 75.3 | 61.5 | 4 | | | | | | | |
| CST | * 448 | 409 | 389 | 91.3 | 86.8 | 4 | | | | | | | |
| XCST | * 640 | 640 | 618 | 100.0 | 96.6 | 4 | | | | | | | |
| PCB | * 130 | 100 | 79 | 76.9 | 60.8 | 21 | | | | | | | |
| IOQ | * 600 (594) | 89 | 53 | 14.8 | 8.8 | 12 | | | | | | | |
| DSK10 | * 383 (368) | 370 | 6553 | OV | 9 | 6 | | | | | | | |
| ATPTBUF* | 317 (317) | 51 | 28 | 16.1 | 8.8 | 69 | | | | | | | |
| SBUF | * 100 (98) | 1 | 0 | 1.0 | 00.0 | 4 | | | | | | | |
| TRL | * 96 | 64 | 61 | 66.7 | 63.5 | 4 | | | | | | | |
| SPREQ | * 657 (525) | 7 | 4 | 1.1 | .6 | 6 | | | | | | | |
| ICS | * 4096 | 1065 | | 26.0 | | 1 | | | | | | | |
| CSTBK | * 108 | 73 | 68 | 67.6 | 63.0 | 1 | | | | | | | |
| SWAPT | * 1028 | 580 | 385 | 56.4 | 37.5 | 6 | | | | | | | |
| JPCNT | * 92 | 17 | 17 | 18.5 | 18.5 | 1 | | | | | | | |
| WMEM | * 11260 | 77632 | 60352 | 68.9 | 53.6 | SECTORS | | | | | | | |
| SPOOL | 384000 | 47424 | 47424 | 12.4 | 12.4 | SECTORS | | | | | | | |
| MEMORY | 4096 | 2790 | 2790 | 68.1 | 68.1 | K WORDS | | | | | | | |

IN MEMORY: SYSCODE=42368 USERCODE=722552 SYSDATA=153376 USERDATA=1558916
 CURRENT # SESSIONS=14 # JOBS=3 MAX # OF SESSIONS=14 # JOBS=3
 TIME (IN SECONDS) SINCE START=31 START: 19:10:22 CURRENT: 19:10:53

```

***** System Surveyor B.00.00 (JAK) WED, JUN 11, 1986, 6:51 PM Elapsed 00:00:30
***** C P U ***** ** Paused for Disc IO **** Memory Garbage Collection
Idle   Busy   Cache  Mem  User&Mem  User  Cache  alloc.  Global  Local
1.3%  2.0%  0.7%  0.6%  0.3%  0.0%  0.0%  1.0%  0.0%  0.1%
-----
Drive- 1 2 3 4 5
R/sec- 0.0 0.0 0.0 0.0 0.0
W/sec- 0.0 0.0 0.0 0.7 0.0

CPU: Avg CPU Busy- 0ms; Avg Shrt Trns Time- 88ms; XPreempts- 0
MEM: Launches/Sec- 0; 'Swaps'/Launch- 0.0; Mem Click Rate 47
IO: IOs/Term read- 0; CPU msec/IO- 27164;

-----
Program Name      J/S#  Pin  Q & %Tot  *W A I T S T A T E S*  Disc IOs
UCOP              0  L8  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
SP                0  L12  1%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C23  14%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .S1113  C25  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
DSMONX            .PUB  .SYS  0  L26  19%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C31  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C36  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
CI                S1111  C39  22%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C43  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C52  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT080P            .DPP  .ELDEC  S1113  C53  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C54  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C55  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT050P            .DPP  .ELDEC  S1113  C56  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT080P            .DPP  .ELDEC  S1113  C58  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C59  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C66  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C68  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C70  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
DSMONX            .PUB  .SYS  0  L71  1%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C85  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C87  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
SURVEYOR          .PRV  .TELESUP  S1141  C88  1%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C91  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C103  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C105  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT060P            .DPP  .ELDEC  S1113  C106  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C112  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C116  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C120  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
CI                S1113  C121  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT080P            .DPP  .ELDEC  S1115  C125  1%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
OP1               .PUB  .SYS  J1171  D127  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%
UT030P            .DPP  .ELDEC  S1113  C128  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%  0%

```


TURBOIMAGE PROFILER STATISTICS REPORT

TRACE INTERVAL FILE : DETROIT

BUFFER ACTIVITY

AGGREGATE BY DATABASE

| Database Name | Dataset Number | Buffer Read Hit Ratio | Buffer Write Post Ratio | Modification Percentage | Available I/O Buffers |
|--------------------|----------------|-----------------------|-------------------------|-------------------------|-----------------------|
| | | (%) | (%) | (%) | AVG |
| DBVM.DATACORP.TEST | 04 | | 88 | 14 | .17357 |
| (1 DBOpens) | All DataSets | | 88 | 14 | .17357 |
| | | | | | 10 |
| | | | | | 0 |

TURBOIMAGE PROFILER STATISTICS REPORT

TRACE INTERVAL FILE : DETROIT

LOCK ACTIVITY

AGGREGATE BY DATABASE

| Database Name | Locking Level (UnCond Locks/Cond Locks) | Queueing Frequency | Time To Acquire Lock (msec) | Unconditional | | | Conditional | | | |
|--------------------|---|--------------------|-----------------------------|---------------|-----|-----|------------------|-----------------------|-------|-------|
| | | | | AVG | DEV | MAX | Denial Frequency | Lock Held Time (msec) | AVG | DEV |
| DBVM.DATACORP.TEST | SET (6/ 0) | 0 | 0 | 0 | 0 | 0 | 0 | ***** | 47360 | 55740 |

REPORT OF JOBS/SESSIONS GENERATED FROM SYSTEM LOGFILE ANALYSIS
SORTED BY CPU

| ACCOUNT | JOBNAME | USER | GROUP | YMMDD | HH:MM | J/S | Q | LDEV | I-O | CPU | MINUTES | TERM |
|---------|-----------|----------|----------|--------|-------|-------|------|------|--------|------|---------|-------|
| | | | | | | | | IN | OUT | WALL | | I-O |
| | | | | | | | | ==== | ==== | ==== | ==== | ==== |
| ELDEC | .PA501J | .MGR | .DATAIR | 860609 | 22:04 | J0933 | E 10 | 12 | 118737 | 400 | 15 | 0 |
| ELDEC | .PA503J | .MGR | .DATAIR | 860609 | 22:41 | J0935 | E 10 | 12 | 42461 | 355 | 16 | 0 |
| ELDEC | .O550MCJ | .MGR | .DATACMD | 860609 | 20:55 | J0895 | D 10 | 12 | 1210 | 326 | 28 | 0 |
| ELDEC | .SF550CEJ | .MGR | .DATACMD | 860609 | 21:35 | J0927 | D 10 | 12 | 14248 | 283 | 25 | 0 |
| ELDEC | .PA100J | .MGR | .DATAIR | 860609 | 21:42 | J0929 | E 10 | 12 | 97211 | 227 | 8 | 0 |
| ELDEC | .PA430IRJ | .MGR | .DATAIR | 860609 | 23:28 | J0943 | E 10 | 12 | 65326 | 213 | 9 | 0 |
| ELDEC | .NO-INIT | .NO-INIT | .DATACMD | 860609 | 20:33 | J0846 | E 10 | 12 | 12524 | 202 | 18 | 0 |
| ELDEC | .PA110IRJ | .MGR | .DATAIR | 860609 | 21:50 | J0930 | E 10 | 12 | 17555 | 198 | 7 | 0 |
| ELDEC | .PA595IRJ | .MGR | .DATAIR | 860609 | 22:56 | J0936 | E 10 | 12 | 2192 | 167 | 7 | 0 |
| ELDEC | .PA502J | .MGR | .DATAIR | 860609 | 22:18 | J0934 | E 10 | 12 | 67469 | 161 | 23 | 0 |
| ELDEC | .NO-INIT | .NO-INIT | .PUB | 860609 | 20:32 | J0862 | E 10 | 12 | 2559 | 98 | 6 | 0 |
| ELDEC | .NO670HCJ | .MGR | .DATACMD | 860609 | 20:55 | J0900 | D 10 | 12 | 738 | 89 | 11 | 0 |
| ELDEC | .PA130IRJ | .MGR | .DATAIR | 860609 | 21:11 | J0917 | E 10 | 12 | 3542 | 77 | 14 | 0 |
| ELDEC | .PA435J | .MGR | .DATAIR | 860609 | 23:36 | J0945 | E 10 | 12 | 15294 | 77 | 3 | 0 |
| ELDEC | .AR310SSJ | .MGR | .DPP | 860609 | 20:33 | J0728 | C504 | 504 | 860 | 69 | 54 | 39558 |
| ELDEC | .NO750MCJ | .MGR | .DATACMD | 860609 | 20:40 | J0878 | D 10 | 12 | 1192 | 60 | 3 | 0 |
| ELDEC | .MP870HCJ | .MGR | .DATACMD | 860609 | 21:06 | J0916 | D 10 | 12 | 107 | 60 | 8 | 0 |
| ELDEC | .AR310PCJ | .MGR | .DPP | 860609 | 20:33 | J0729 | C505 | 505 | 662 | 59 | 20 | 27606 |
| ELDEC | .NO560CNJ | .MGR | .DATACEN | 860609 | 20:31 | J0868 | D 10 | 12 | 1543 | 52 | 4 | 0 |
| ELDEC | .PA120IRJ | .MGR | .DATAIR | 860609 | 23:05 | J0942 | E 10 | 12 | 3036 | 52 | 23 | 0 |
| ELDEC | .EM240IRJ | .MGR | .DATAIR | 860609 | 23:00 | J0939 | D 10 | 12 | 16447 | 51 | 4 | 0 |
| ELDEC | .SC080HCJ | .MGR | .DATACMD | 860609 | 20:46 | J0884 | D 10 | 12 | 589 | 48 | 4 | 0 |
| ELDEC | .PA030IRJ | .MGR | .DATAIR | 860609 | 21:31 | J0922 | E 10 | 12 | 7813 | 44 | 4 | 0 |
| ELDEC | .EM290IRJ | .MGR | .DATAIR | 860609 | 23:04 | J0941 | D 10 | 12 | 4702 | 44 | 4 | 0 |
| ELDEC | .EM230IRJ | .MGR | .DATAIR | 860609 | 22:56 | J0937 | D 10 | 12 | 1543 | 42 | 5 | 0 |
| ELDEC | .PA371IRJ | .MGR | .DATAIR | 860609 | 21:35 | J0926 | D 10 | 12 | 8690 | 40 | 3 | 0 |
| ELDEC | .SF591CEJ | .MGR | .DATACEN | 860609 | 21:37 | J0928 | D 10 | 12 | 515 | 39 | 2 | 0 |
| ELDEC | .NO560MCJ | .MGR | .DATACMD | 860609 | 20:54 | J0905 | D 10 | 12 | 619 | 38 | 5 | 0 |
| ELDEC | .NO560MCJ | .MGR | .DATACMD | 860609 | 20:46 | J0885 | D 10 | 12 | 625 | 37 | 5 | 0 |
| ELDEC | .PA020IRJ | .MGR | .DATAIR | 860609 | 21:02 | J0911 | E 10 | 12 | 4226 | 36 | 5 | 0 |
| ELDEC | .SC200SSJ | .MGR | .DPP | 860609 | 21:29 | J0739 | C504 | 504 | 216 | 33 | 22 | 11438 |
| ELDEC | .EN080MCJ | .MGR | .DATACMD | 860609 | 20:50 | J0899 | D 10 | 12 | 1558 | 32 | 4 | 0 |
| ELDEC | .NO700MCJ | .MGR | .DATACMD | 860609 | 20:38 | J0876 | D 10 | 12 | 1082 | 29 | 2 | 0 |
| ELDEC | .RO200HCJ | .MGR | .DATACMD | 860609 | 20:43 | J0886 | D 10 | 12 | 503 | 26 | 3 | 0 |
| ELDEC | .EN070HCJ | .MGR | .DATACMD | 860609 | 20:47 | J0887 | D 10 | 12 | 448 | 23 | 3 | 0 |
| ELDEC | .PA010J | .MGR | .DATAIR | 860609 | 21:01 | J0910 | E 10 | 12 | 1733 | 21 | 2 | 0 |
| ELDEC | .AR310MCJ | .MGR | .DATACMD | 860609 | 20:43 | J0889 | D 10 | 12 | 1177 | 20 | 2 | 0 |
| ELDEC | .AP200MCJ | .MGR | .DATACMD | 860609 | 20:52 | J0897 | D 10 | 12 | 632 | 18 | 2 | 0 |
| ELDEC | .IRENSTJ | .MGR | .DATAIR | 860609 | 21:35 | J0926 | E 10 | 12 | 519 | 18 | 7 | 0 |
| ELDEC | .AP180J | .MGR | .DATAIR | 860609 | 20:34 | J0871 | D 10 | 12 | 1352 | 17 | 2 | 0 |
| ELDEC | .SPOCCOPY | .MGR | .DPP | 860609 | 21:07 | J0734 | C505 | 505 | 613 | 17 | 8 | 453 |

| TYPE COMMAND | Tenth's of a Sec | | COUNT | Tenth's of a Second | | CPU |
|-----------------|------------------|-------|-------|---------------------|-----|-----|
| | WALL | CPU | | WALL | CPU | |
| COMD APAHDD | 1259 | 179 | 47 | 26 | | |
| COMD APALTD | 10554 | 586 | 324 | 32 | | |
| COMD APATRD | 245 | 3 | 79 | 3 | | |
| COMD APCCXD | 50 | 11 | 2 | 25 | | |
| ORIG CEN | 83422 | 9563 | 4820 | 17 | | |
| ORIG CORP | 4282 | 1181 | 1052 | 4 | | |
| ORIG MCD | 33989 | 4974 | 2328 | 14 | | |
| ORIG PCD | 120271 | 17329 | 6352 | 18 | | |
| ST-% 10 | 1 | | 1536 | | | |
| ST-% 20 | 1 | | 3071 | | | |
| ST-% 30 | 3 | | 4607 | | | |
| ST-% 40 | 6 | | 5142 | | | |
| ST-% 50 | 10 | | 7678 | | | |
| ST-% 60 | 12 | | 9213 | | | |
| ST-% 70 | 17 | | 10748 | | | |
| ST-% 80 | 23 | | 12284 | | | |
| ST-% 90 | 34 | | 13819 | | | |
| ST-% 100 | 3041 | | 15354 | | | |
| SYS *-BAP | 272351 | 33328 | 17 | 10 | | 1 |
| SYS *-BBAP | 103992 | 17720 | 24 | 12 | | 1 |
| SYS *-TOT | 376343 | 51048 | 19 | 10 | | 1 |
| SYS A-BAP | 83218 | 10762 | 4758 | 17 | | 1 |
| SYS A-BBAP | 38151 | 6863 | 1654 | 22 | | 18 |
| SYS A-TOT | 121369 | 17625 | 6422 | 18 | | 13 |
| SYS *TOTAL* | 376343 | 51054 | 19546 | 19 | | 1 |
| TERM A027 | 10081 | 74 | 109 | 92 | | 109 |
| TERM A030 | 9137 | 1356* | 431 | 21 | | 28 |
| TERM A031 | 1968 | 329 | 69 | 4 | | 4 |
| TERM A033 | 1226 | 154 | 267 | 20 | | 20 |
| TIME 8606100618 | 4447 | 594 | 212 | 17 | | 17 |
| TIME 8606100700 | 19160 | 3456 | 1111 | 19 | | 19 |
| TIME 8606100800 | 29755 | 4412 | 1516 | 18 | | 18 |
| TIME 8606100900 | 35840 | 5548 | 1890 | 19 | | 19 |
| TIME 8606101000 | 45718 | 4726 | 2340 | 15 | | 15 |
| TIME 8606101100 | 23740 | 2654 | 1569 | 17 | | 17 |
| TIME 8606101213 | 26745 | 3069 | 1535 | 25 | | 25 |
| TIME 8606101300 | 56709 | 8074 | 2232 | 24 | | 24 |
| TIME 8606101400 | 64647 | 9643 | 2877 | 19 | | 19 |
| TIME 8606101500 | 48653 | 5596 | 2524 | 63 | | 63 |
| *TOTAL* | 376343 | 51054 | 19546 | 19 | | 19 |
| USER A1049 | 6222 | 368 | 98 | 11 | | 11 |
| USER A3850 | 341 | 12 | 30 | 22 | | 22 |
| USER A4064 | 341 | 93 | 15 | | | |

<< The above listing contains sample entries for each type logged >>

APPENDIX B -- CHAIN ELONGATION DESCRIPTION & DISCUSSION

The term "chain elongation" is used to describe a condition which occurs when accessing information in a data base using a key value chain requires more disc I-O than is theoretically required if all entries on that chain were packed such that reading the chain resulted in a minimum number of disc blocks being read. To best understand how elongation occurs, following are several examples demonstrating how elongation might occur for a chain with the key value of "X". The following symbols are used to facilitate understanding:

- * = unused record
- X = Record of the chain being studied
- a,b,c,d,e,f = records containing chains of keyvalue a,b,c,d,e and f values respectively
- | = block separator

Case 1 -- Elongation = 1.00 (Optimum)

| BLK 1 | BLK 2 | BLK 3 | BLK 4 | BLK 5 | BLK 6 | BLK7 | BLK 8 |
|--------|-------|-------|-------|-------|-------|-------|-------|
| aaXXX* | ***** | ***** | ***** | ***** | ***** | ***** | ***** |

The three records in the chain X are all in the same block (BLK 1) and therefore the elongation is 1.00 since the expected number of blocks is 1 (# records/block factor).

Case 2 -- Elongation = 1.00 (Optimum)

| BLK 1 | BLK 2 | BLK 3 | BLK 4 | BLK 5 | BLK 6 | BLK7 | BLK 8 |
|--------|--------|--------|-------|-------|-------|-------|-------|
| aabbbX | XXXXXX | ccc*** | ***** | ***** | ***** | ***** | ***** |

The expected number of blocks is 2 (7 records divided by 6 records/block). Since all records of the chain X are contained within two blocks, then the elongation is 1.00 (actual blocks/expected blocks).

APPENDIX B -- CHAIN ELONGATION DESCRIPTION & DISCUSSION(CONTINUED)

Case 3 -- Elongation = 1.50

| BLK 1 | BLK 2 | BLK 3 | BLK 4 | BLK 5 | BLK 6 | BLK7 | BLK 8 |
|--------|--------|--------|-------|-------|-------|-------|-------|
| aabbXX | XcccXX | XXddd* | ***** | ***** | ***** | ***** | ***** |

The expected blocks is two (7 records divided 6 records/block). the actual number of blocks to contain the chain is three. Thus the elongation for chain X is 1.5 (3 actual blocks/ 2 expected blocks).

Case 4 -- Elongation = 6.00

| BLK 1 | BLK 2 | BLK 3 | BLK 4 | BLK 5 | BLK 6 | BLK7 | BLK 8 |
|--------|--------|--------|--------|--------|--------|-------|-------|
| aaXaaa | bbbbXb | ccccXc | dddXdd | eeeeXe | Xfffff | ***** | ***** |

The elongation for chain X is 6.00 since the expected number of blocks for this chain is 1.00 (6 records divided by 6 records/block).

Note that elongation is a function of reading more blocks than is required based upon the expected number of blocks being equal to the number of records divided by the blocking factor. Case 5 shows that the actual number of blocks read is based upon the chain linkage.

Case 5 -- Elongation = 3.50

| BLK 1 | BLK 2 | BLK 3 | BLK 4 | BLK 5 | BLK 6 | BLK7 | BLK 8 |
|--------|--------|-------|-------|-------|-------|-------|-------|
| aaaXXX | XXXX** | ***** | ***** | ***** | ***** | ***** | ***** |

```

| | | | | | | |
| | | | | | | |Record 10
| | | | | | | |Record 9
| | | | | | | |Record 8
| | | | | | | |Record 7
| | | | | | | |Record 6
| | | | | | | |Record 5
| | | | | | | |Record 4

```

APPENDIX B -- CHAIN ELONGATION DESCRIPTION & DISCUSSION(CONTINUED)

In this case, the actual chain linkage for key X is from record 10 to record 4 to record 9 to record 5 to record 8 to record 6 to record 7. Since records 7,8,9 & 10 are in block 2 and records 4,5 & 6 are in block 1, this means that to read all records in the chain X, the actual blocks read were #2, #1, #2, #1, #2, #1 & #2. Thus 7 blocks reads were required to access all the seven records in the chain. Since the expected number of blocks was 2, then the elongation in Case 5 is 3.5 .

The type of distribution in Case 1 and Case 2 is typical of a system in which a user locks a data set, adds several records on a key item and then unlocks the set. Such distribution is also common when a user adds records to a set for which little contention exists.

The type of distribution in Case 3 and Case 4 is typical of situations where a user add records to a set with high activity or over extended time periods. For example, Case 4 shows a situation where chain "a" had two records added, then chain "X" had one record added, then chain "a" had three more records added, then chain "b" had four records added, then chain "X" had a second record added, then chain "b" had a fifth record added , etc.

Case 1 through Case 4 are all typical of sets without a delete chain. Case 5 is typical of a set where records have been deleted or where the chain is sorted on another item in the record which doesn't correspond to the order in which the records were added. Suppose the following situation had existed prior to the adding of chain "X" to Case 5:

| BLK 1 | BLK 2 | BLK 3 | BLK 4 | BLK 5 | BLK 6 | BLK7 | BLK 8 |
|--------|--------|-------|-------|-------|-------|-------|-------|
| aaabcd | efgh** | ***** | ***** | ***** | ***** | ***** | ***** |

Now suppose that records were deleted in the following sequence: e,d,f,c,g,b,h. When the "X" records were added, they would be added in the following sequence: h,b,g,c,f,d,e. Thus if one were to read the "X" chain, one would read record 4 of block 2(h), then record 4 of block 1(b), then record 3 of of block 2(g), record 5 of block 1(c), etc.

The order of reading a chain which is sorted is the order of the value of the sort field. If the sort field for two or more records is equal, then any field(s) following the sort field will be used in an extended sort to order the records. Thus one can see that reading a sorted chain could easily result in records being read in a non-sequential order and in fact could cause the re-reading of blocks of information as shown in Case 5.

IMPACTS OF TECHNOLOGY ON HIGH-PERFORMANCE MASS STORAGE

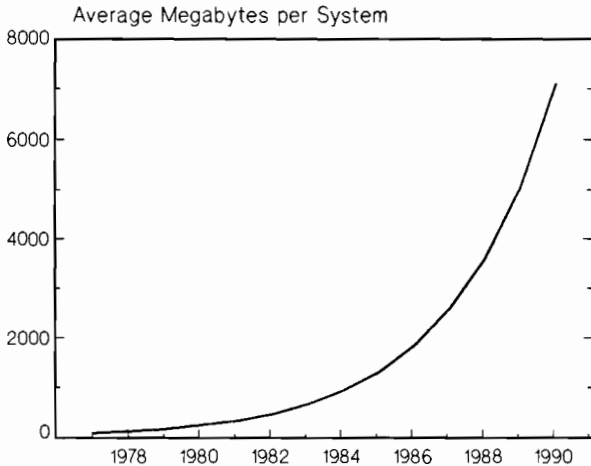
David R. James
Hewlett-Packard
P.O. Box 39
Boise, ID U.S.A. 83707



Ten years ago, few of us could have envisioned holding 50 Megabytes of disc storage in the palm of our hand. Even fewer could have imagined picking up the phone and buying those 50 Megabytes with a Visa card. Such a disc drive weighed several hundred pounds and cost the equivalent of several cars. Some of us still have a 50 Megabyte HP 7920 around; they still work fine, their performance is good, and they remind us where we came from.

Correspondingly, few of us envisioned so many people with direct access to computers. Now nearly everyone seems up on the latest computer development, and they want more of what the machines hold--information. Bulging computer rooms also show the result of the demand for this information. Many data centers seem to swim in a sea of disc drives. Advances in technology give back some of that space, but often not rapidly enough to save the office next to the computer room. The following chart shows the growth in demand for on-line information storage.

Information Storage Growth
HP 3000 Systems



Hewlett-Packard, as a disc drive manufacturer, has simply to follow the rule of thumb: make disc drives with higher capacity, better performance, smaller size, better reliability and lower cost. To do this contributions from several engineering disciplines are required. It is the purpose of this paper to examine those technologies and how they impact high performance mass storage.

Heads, Media and More

Heads and media are the heart of a disc drive. The interface between them is "where the rubber meets the road." The race is to increase the areal density fast enough to keep up with rapidly growing demand and still stay within a not so rapidly growing data processing budget. Hand in hand with the demand for a larger quantity of data is the demand for that data to be available at all times. Reliability has moved to the "top of the charts" of computer owners' expectations.

To increase areal density, bits per inch (bpi) and tracks per inch (tpi) must be increased. This is done by decreasing the gap of the head, flying the head closer to the surface and improving the physical and magnetic properties of the media. Along with these fundamental changes in heads and media, better servo techniques are used, the mechanical structures are refined and the electronics that convert analog to digital signals must be improved. As design trade-offs are made to develop these new devices, reliability is the foundation for decision making. New technologies are not employed until they are proven. Testing is a major part of the design cycle and the manufacturing process is developed in unison with the drive. The result is a high quality, high capacity disc drive.

Thin Film Media

Higher areal densities are accomplished through shorter flux transitions. This requires a reduced read/write head gap and a smaller distance between the head and media. Additional increases in density are possible as the magnetic properties of the media are improved. Oxide media has been a standard since the beginning of the disc drive industry. In recent years, however, thin-film media has become a viable choice.

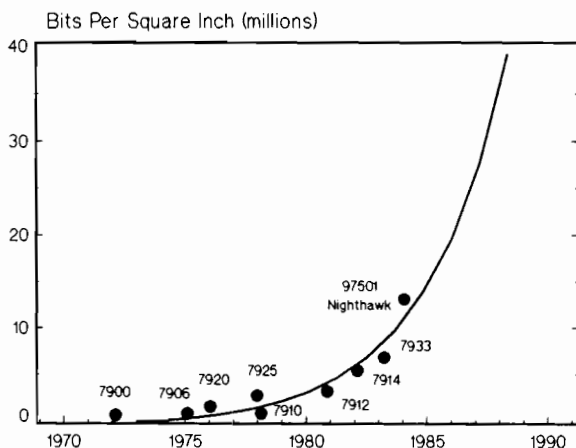
Conventional particulate coatings are based on gamma ferric oxide particles. These particles are suspended in a non-magnetic binder that is spin coated onto a polished aluminum

Impacts Of Technology On High-Performance Mass Storage

substrate. The resulting coat is 20 to 40 microinches thick and is capable of allowing bit densities of 10,000 to 15,000 bpi.

Thin film media offers higher density, improved signal to noise ratios and a hard, smooth surface. The coatings are either sputtered or plated onto the substrate. The media produced from this process has a coat that is two to five microinches thick and is capable of supporting bit densities well above 15,000 bpi. The following graph illustrates advancements seen in areal density.

Growth in Areal Density



As mentioned before, an advantage of thin-film media is the extremely smooth surface. Conventional media allows head flying heights from 14 to 19 microinches. Thin film media allows flying heights as low as six to eight microinches. Thin-film media also offers a signal-to-noise ratio of 2:1 times better than conventional media. The result is a stable base to substantially increase areal density.

The hardness of the media surface makes thin film media less prone to damage during shipment and resistant to head crashes during operation. When a head bounces on oxide media, particles are jarred loose. Significantly greater shock must occur with thin film media before similar problems arise.

HP has invested heavily in sputtered thin film media research and development. Sputtered technology allows greater control of the magnetics. The sputtered coat also eliminates the use of lubricants, reducing the chance that heads will stick to the landing zone.

Thin Film Heads

Over time, areal density has increased from 2000 bits per square inch to over 22 million bits per square inch. Head technology has accounted for much of this gain. Recording head gaps decreased from 1000 microinches to 25 microinches. Flying heights plunged from 800 microinches to less than 10 microinches.

Magnetic read/write heads consist of three major parts: the actual head, a slider to hold the head and give it aerodynamic properties and a flexure to extend the head out from the arm assembly. Our primary focus will be the head.

Ferrite heads are manufactured from extruded magnetic poles with hand wound coils. The size of the poles, the width of the gap and the number of windings in the coils determine the magnetic properties. These heads require machining to bring them into specific dimensions. Ferrite heads can offer track densities of 1000 tracks per inch.

The machining process proves to be a major limiting factor in ferrite head manufacturing. One approach to overcoming this limitation is to bond the ferrite core in glass before machining. This is referred to as a composite head. This results in heads capable of 1500 tracks per inch. While this technique is an improvement, ferrite heads are still limited.

Thin film heads first made their appearance in the market in 1979. These heads are manufactured using production methods taken from the semiconductor industry. A semiconductor wafer is used in the manufacturing process with thin layers (or films) of material deposited on it. These layers create the equivalent of a ferrite head's pole, gap and winding. One semiconductor wafer can produce as many as 500 thin film heads. The manufacturing cost of a thin film head is less than the manufacturing cost of a conventional head. However, to implement this technology requires large start-up funds. Until these initial manufacturing costs are depreciated or volumes increase significantly, the price vendors charge for a thin film head remains high.

A thin film head has significantly smaller mass, improving its aerodynamic qualities. The manufacturing process allows greater control of the magnetic characteristics of the head. This allows the head to fly closer to the surface and again, improve areal density. Thin film heads offer track densities of 2000 tracks per inch.

Disc Drive Mechanisms

As the technology for heads and media improves, corresponding improvements are required in the disc drive mechanism. These improvements fall into two categories: servo and disc drive mechanical assemblies.

Servo systems were incorporated into disc drives as track densities increased beyond the capabilities for the mechanism alone to accurately position the head over the track. These systems provide control feedback during head movement and track following feedback after the head has settled on track. Beyond providing feedback, the servo system addresses two problems that may appear in the mechanism: register alignment and track runout. Register alignment problems can occur when temperature changes or thermal gradients shift the relationship between the spindle and the actuator. Track runout occurs when the data track on a particular surface is no longer centered relative to the axis of rotation. In disc drives with high track density, minute changes in the alignment of the mechanism have potentially severe consequences. Advanced servo systems effectively compensate for these changes and assure reliable operation.

Fundamental design of the mechanical system in the disc drive greatly impacts the capacity, performance and reliability. Optimizing performance and cost in a high precision environment requires substantial skill. Hewlett-Packard is currently in the design phase of its eighth generation of disc drives. A substantial skill base has been built in design, manufacturing and management. This skill base provides products on the leading edge in reliability, performance and cost.

Optical Discs

In the past few years, a great fervor has arisen in the industry over optical discs. Optical discs come in several different varieties, but always carry a common characteristic--the capability of significantly improving areal density. Three classes of optical discs currently exist:

Optical Read Only Memory (OROM); Write-Once, Read-Many (WORM) and Erasable Optical disc.

The OROM has recently come out of the lab and appeared on the market. With this product, the media is written or stamped at a factory. The OROM is useful for storing and distributing large quantities of stagnant information. A unique feature of an OROM is that the media can be produced in a stamping process. Large quantities of data can be written on a disc nearly instantaneously. This is in contrast with other types of magnetic memory currently being used where the data is written serially.

WORM drives permanently change the surface of the media during the write process. The fundamental approach to recording with this drive is to change the media in such a way that it will in turn consistently alter a laser beam directed at the media. Three approaches are typically used:

Pit forming or ablative - in this approach, a pit is formed in the active layer by a short laser burst that imparts sufficient energy to melt a pit in the material. The pit has a diameter of about one micron.

Bubble forming - in this technique, the heat from a laser partially vaporizes the layer of material directly beneath the active material. The active material deforms into the shape of a bubble from the gaseous pressure beneath.

Phase change - here, the active layer can exist with stability in two different states with differing reflectivity. With the application of a very rapid laser pulse, the material is heated just above melting point. As the material cools it changes states with a corresponding change in reflectivity.

The U.S. government has determined that media recorded with WORM devices is one of the few acceptable types of information storage usable in a court of law. For this reason, write-once drives are viewed as a valuable peripheral for archiving data.

The final type of optical disc currently being researched is an erasable optical disc. These discs also use a laser but they write on magnetic media. The laser raises the temperature of the media in a limited area and decreases the coercivity at that point. A weak magnetic field is present which changes the orientation of the magnetic domain.

Impacts Of Technology On High-Performance Mass Storage

Erasing is accomplished by reversing the orientation of the magnetic domain.

All three of these optical discs currently exhibit relatively low performance. Most OROM and WORM optical disc vendors have devices with 200 Megabytes to several gigabytes and floppy disc-type performance. Erasable optical disc offers the potential of higher performance but with the penalty of a separate erase cycle. This results in a requirement for an additional rotational latency, thus slowing performance.

In an optical disc, the distance between the head and the media is several orders of magnitude greater than in a conventional magnetic disc. This distance greatly reduces the susceptibility to dust and the possibility of head crashes. These greater distances also make it easier to retract the heads and free the media for removal from the drive. Removability is important to many applications, particularly archiving.

Error rates are currently a concern with optical disc. The high bit density has come at the expense of raw error rates. Sophisticated error correction codes must be developed and employed in these devices. These codes are beginning to be available, but it will take further improvement to bring error rates into acceptable levels.

So the question remains, "When will I see an optical disc and will optical discs replace my magnetic discs?". Optical discs are slowly appearing on the market today. Currently, several major vendors are selling read only discs. It is projected that in the next several years, write once optical discs will be offered in many computer vendors' product lines. However, these discs will act as a supplementary mass storage medium to current magnetics as opposed to replacement for current magnetic discs. Erasable technology remains on the three-to-four year time horizon.

As engineers pursuing optical disc technologies close in on their goals and solve more and more of their problems, engineers working on magnetic discs continue to move forward with improvements in that technology as well. We currently project that magnetic discs will be able to maintain cost effectiveness through the 1990's. Hewlett-Packard is investing in both of these areas to ensure that the benefits from each can be realized on HP computers.

Impacts Of Technology On High-Performance Mass Storage

Leveraged Technologies

Heads, media and optical discs are technologies being developed specifically for mass storage products. Other technologies are also required for successful design of mass storage devices. They include VLSI electronics, fiber optics, advancements in firmware technology, and advancements in software technology.

VLSI electronics are extremely important in advanced disc drive design. The capability of shrinking an entire PC board down to the size of a single chip not only lowers the cost of those electronics, but improves reliability and increases performance. Often times, the performance of a disc drive is constrained by the electronics. Higher bit densities mean higher transfer rates, if the disc spins at the same speed. If the controller or read/write electronics cannot handle the higher transfer rates, disc rotation rate must be slowed with a corresponding decrease in overall disc performance.

Hewlett-Packard is investing heavily in very high performance electronics. With current high speed NMOS technology and promises for development in CMOS that will carry that performance orders of magnitude beyond what we see today, the future for electronics in disc drives is bright.

As data comes out of a disc subsystem faster, the required buses to transport that data to the computer become a key performance factor. Performance can be increased on current interfaces; however, cable lengths are shortened and/or the cost increases. Here the use of fiber optics holds much promise. These cables (about the same size as the common lamp cord) can carry data between the disc and the computer several times faster than buses currently used. Fiber optics do not have the types of length restrictions that copper cables have. They are also immune to any type of electro-magnetic interference, improving the integrity of the data as it moves from the disc to the computer. Hewlett-Packard has been producing high quality fiber optic components for many years and is seen as a leader in the industry. This leadership position is seen in terminal connections today and will be used for disc connections as well.

Software and firmware are key links in taking SPUs and disc drives and creating a system. Hewlett-Packard leads the industry with advanced protocols and extensive on-line diagnostics in the early '80s. That leadership position

continues to be advanced in development labs where software technology will improve the reliability, performance and functionality of mass storage subsystems.

As the demand for information storage on HP systems increases, the investment being made by Hewlett-Packard will continue to pay off for you with higher reliability, better performance and lower cost.

In summary, the following table details the benefits offered through HP's developments in mass storage technology.

| <u>Technology</u> | <u>Benefit</u> |
|----------------------|---|
| Thin film media | Higher capacity and lower cost More reliable |
| Thin film heads | Higher capacity and lower cost |
| Disc mechanisms | Higher performance More reliable |
| Optical discs | Higher capacity and lower cost Removable media |
| VLSI electronics | Higher performance Lower cost |
| Fiber optic channels | Higher performance Longer cable lengths |
| Firmware | Higher performance More reliable |

DJ/ad
dj

IMPROVING YOUR PERFORMANCE

R.E. VAN VALKENBURGH
AMPEX CORPORATION
P.O. BOX 190
MARVYN PKWY.
OPELIKA, AL U.S.A 36803-0190

The larger HP3000 systems in use today for standard business and manufacturing applications are requiring a much more sophisticated approach in management to achieve an acceptable level of performance. We will examine some characteristics that relate to larger systems which tend to be I/O intensive with predominant use of IMAGE. We will dispell some common myths relating to the various subsystems, and eventually proceed to developing a response curve for a particular data base system.

INTRODUCTION

Particularly in recent times the workload capabilities of HP3000's have been increased many-fold. The advent of the series 6X, 7X, MPE-V, Turbo-IMAGE, improvements to I/O hardware, and next the "spectrum machines" have or will be contributing to far greater capabilities and creating far greater diversity in the workloads. For this reason we may now need to take a closer look at the old "rules of thumb" that we've been accustomed to using in operating our collective HP3000's.

The intent of this paper is to provide some insight into the performance characteristics of the highly interactive real-time business and manufacturing workloads that are, or have been, growing along with (ahead of) the HP3000. Frequently the major goals of MIS departments include maximizing resource utilization and minimizing response time. This has become a much more difficult task as our systems have grown, and as we shall be seeing in the course of this paper, MAXIMIZING RESOURCE UTILIZATION AND MINIMIZING RESPONSE TIME ARE COMPETING GOALS AND ARE FREQUENTLY MUTUALLY EXCLUSIVE on interactive systems. (Take that to your board of directors).

At times we might be somewhat liberal in our assumptions. Some such assumptions will relate to the most extreme of situations. Often understanding well the extremes (which are frequently "clear cut") is sufficient conceptually, and the conceptual transition to the less extreme "real-life" is trivial. I think you will in time agree that due to the large amount of information being touched on here, this approach will be most productive.

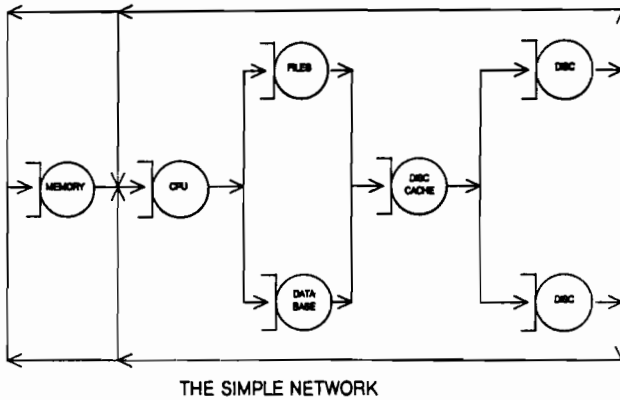
CONTENTS

| | |
|--|---|
| INTRODUCTION | * SOFTWARE DISC CACHING, a SHARED RESOURCE |
| 1. OVERVIEW, the NETWORK | Costs of Disc Caching |
| * The CPU, a SERIALLY REUSEABLE RESOURCE | Additional Caching Tips |
| Priority Preemption | * IMAGE, a SERIALLY REUSEABLE RESOURCE |
| Dynamic Reprioritization | IMAGE and the File System |
| Transaction Orientation | IMAGE and Software Disc Caching |
| Temporary Reprioritization | Synonyms |
| Service Times | Clustering |
| Process Priorities and Actual Scheduling | Hashed Access |
| * MEMORY, a SHARED RESOURCE | Detail Data Sets |
| * DISC I/O and the FILE SYSTEM | Data Set Control Record |
| Maximizing Contiguously Located Transfers | IMAGE Block Sizes and Buffers |
| Achieving Balanced I/O Across Disc Drives | Comments on "Single Threading" |
| Further Concurrency of I/O | Predicting Performance of IMAGE |
| Reduction of Disc Seek Time | * OTHER RESOURCES |
| Reduction of Disc Accesses | 2. QUEUEING THEORY and QUEUEING NETWORK ANALYSIS |
| Buffered Disc Access | 3. An APPLICATION UNDER THE MICROSCOPE |
| Choosing Number of Buffers | 4. CACHING REVISITED |
| Sharing of Buffers | CONCLUSION |
| Unbuffered Disc Access | |

While recognizing that the title of this paper is "IMPROVING YOUR PERFORMANCE", much of the material presented here is background information. The most important part of performance improvement begins with an adequate understanding of the characteristics of the system, without this understanding, performance improvement is at best a haphazard task, and the consequence of our efforts are likely to be a disappointment.

1. OVERVIEW, the NETWORK

It is useful to picture an overall system by way of a network flow diagram. The system is composed of a network of resources, each shared resource having its own queue. Processes enter the system, make various transitions between (visits among) the resources, and finally exit the system when the transaction or job is complete. For all practical purposes here we can consider transitions and service time to be essentially random.



Each resource in this diagram happens to be a shared resource, and each contains a queue for "holding" those processes which are awaiting the resource. Each time that a process arrives for service at a certain resource when the resource is busy, that process will either queue for the resource or preempt the current process (causing that process to be re-queued). One point that becomes very clear in the network diagram is, the rate of flow through a certain path in the network will be restricted to that of the slowest and/or busiest resource (the "bottle-neck").

In general, a typical process will visit the cpu, then visit another resource, visit the cpu, visit another resource, etc. We classify resources as:

- 1) Serially reuseable: only one process at a time can use the resource;
- 2) Concurrent: more than one process may share the resource concurrently; and
- 3) Consumable: the resource is used a limited number of times by a limited number of processes (e.g. a semaphore between specific processes).

It is important that we note some of the characteristics of the most important resources in the network of queues. It is well known that the most limiting resource is the "bottleneck", but what is not as well appreciated is that THE OVERALL PERFORMANCE OF THE SYSTEM WILL TAKE ON THE CHARACTERISTICS OF THE MOST LIMITING RESOURCE (at least in single

jobclass systems). When this point is understood, the nature of many performance problems becomes very clear.

The CPU, a SERIALY REUSEABLE RESOURCE

On the HP3000 the cpu is a serially reuseable resource, as there is never more than one process actively using the cpu at any given instant. If there is more than one process requiring the cpu, all except one will be queued.

The HP3000 schedules processes for the cpu by a prioritization scheme. The operating system maintains a queue for the cpu. Within this queue are several "subqueues", a master queue (which I'll refer to as the M queue), a foreground queue (the C queue), and two background queues (the D and E queues).

In general processes in the M queue have fixed priorities, while processes in the C, D, or E queues may have dynamically changing priorities. The priority range of the M queue is fixed by the operating system, while the range of priorities of the C, D, and E queues may be adjusted by the "tune" command. The range of available priorities to the C, D, or E queues must lie within the range of the M queue. It is the intention (and the default) that processes in the priority range of the C queue are higher in priority than processes in the D subqueue, and likewise the D subqueue over the E subqueue.

Priority Preemption

The scheduling of the cpu is done by priorities. Any process in the M or C queues becoming ready for the cpu will preempt the current process using the cpu if that process is of a lower priority, D or E queue processes, however, will not preempt processes in their respective queues when becoming ready for the cpu, even if a higher priority.

Dynamic Re-Prioritization

Priorities of processes in the C, D, and E queues may be dynamically adjusted by the operating system. When a C, D, or E queue process is initially launched it is given the highest priority permitted for that particular queue. Each time the process within that queue consumes more cpu time than the current "filter" value for that queue it will be dropped in priority (but never lower than the limit for the particular queue). For D and E queues the filter value is specified in the tune command (by the "maxquantum" parameter for the D queue). For the C queue, the filter value may be constantly changing, as it is dependent upon the C queue "average short transaction" time. This average short transaction time is recomputed each time a C queue process has consumed the filter value in cpu time and is a candidate for reprioritization; the average short transaction time is calculated by averaging the cpu time consumed by previous C queue processes for a single transaction, but if that transaction time is outside of the minquantum and maxquantum

parameters specified in the "tune" command, the nearest of the minquantum or maxquantum is used for purposes of recomputing the average. By default, and all other things being equal, a process in the D or E subqueues will be permitted to consume about 3 1/3 times as much cpu than a C queue process, before the process is a candidate for reprioritization.

Transaction Orientation

Processes in the C, D, or E queues will be raised in priority to the highest available in their respective queue any time a terminal read is performed. This implements an interactive "terminal transaction" scheme.

Temporary Reprioritization

There are certain cases in which the scheduling is temporarily modified by a software resource. At present, if a low priority process has obtained a SIR or data base and a higher priority process requests the same resource, the low priority process will temporarily be raised to that of the higher priority (in the IMAGE case it will never exceed the highest priority allowed to C queue processes).

Service Times

Cpu service times of processes of course are extremely variable, and depend upon the nature of the requirements of the particular process as well as the requirements of the currently executing process.

Process Priorities and Actual Scheduling

The priority scheme used by the scheduler is intended to favor interactive processes and processes with small cpu requirements (short transactions). It will be most effective when the cpu is the system bottleneck, however, if the system's (or certain jobclass') bottleneck is other than the cpu, THE SCHEDULER'S PRIORITIZING SCHEME IS LIKELY TO HAVE VERY LITTLE EFFECT on the actual assignment of resources.

MEMORY, a SHARED RESOURCE

Historically the use of main memory on multiprogramming systems has been difficult to analyze. Strictly speaking it is a shared resource, but the nature of the resource is such that it may limit the degree of multiprogramming, that is it may limit the number of processes that may be requiring service by other resources at a given time.

On the HP3000, memory management is handled in part both by the microcode and the operating system. Memory management is handled in terms of segments. With the exception of special "bank 0" areas maintained by and for MPE, all other addressable areas of memory are separable seg-

ments. Some segments are locked into memory for quick access, most segments, however, are "swapped" in and out of memory dependent upon memory demand. There are now three types of segments: data segments, code segments, and cache domains. Many of the system tables' data segments are locked into memory and will never be "swapped". Some of the operating system code segments, defined as "core resident", will likewise never be swapped.

Most of the segments maintained by MPE on behalf of a user process will be swapped in and out of memory as needed (actually a code segment is never swapped out of memory, as its contents are not modified in the course of execution).

Any time a process addresses an absent code or data segment the micro-code detects its absence and causes an "absence trap". The absence trap causes the memory manager (dispatcher) to be invoked to resolve the memory absence as well as to dispatch another ready process, if possible. (Cache domains are not known to the micro-code. Their management is handled entirely by the software, but in a similar manner to code and data segments).

The strategy that the memory manager uses to determine which segments remain in memory and which segments are swapped is based upon a very simple "recent use" algorithm (not a "least recently used" algorithm). Each time a segment (code, or data) is referenced a bit is set by the micro-code (caching software sets this bit for cache domains) indicating that it has been recently accessed. Each time the memory manager is cycling through memory, it checks this bit. If the access bit is off (not recently accessed) and it has not already been swapped it will now be set up for swap. If the bit is set (has been recently accessed) it will be reset by the memory manager. Therefore if the memory manager is able to cycle twice past a particular segment before a process has a chance to access it, it will be swapped. If the memory manager cycles a third time past a certain segment the region will be marked available.

In addition to handling the swapping of segments as needed, the memory manager is responsible for performing housekeeping chores otherwise known as garbage collection. There are two different sets of rules the memory manager follows with respect to garbage collection, local and global garbage collection. Local garbage collection is performed any time a segment is set to be swapped out and the free space is not large enough to contain the new segment. Local garbage collection consists of moving all non-frozen and non-locked segments to the outer edges of their current bank. Frozen segments are those that are flagged as currently "non-swappable" and not permitted to be moved, since the I/O systems use absolute addresses, a target buffer can not be permitted to be moved. Locking a segment in memory indicates that it cannot be swapped out, but allows it to be moved as determined by the memory manager. Code segments are frequently locked into memory to ensure that any attempt to execute that code will not need to await a swap.

Global garbage collection is done only when the memory manager has determined that there is a general "memory supply crisis" and the cpu otherwise has nothing else to do (is paused). The memory manager decides that there is a "memory supply crisis" whenever the time it takes to cycle through memory looking for free space is less than the "min-clockcycle" setting of the "tune" command. Global garbage collection is an attempt by the memory manager to increase the size of the free areas from largest to smallest.

Locked and frozen segments (especially frozen segments) make the work of the memory manager much more difficult. When there is a large amount of I/O being performed on a system there will tend to be a large number of frozen segments, which tend to diminish the value of local garbage collection, and consequently increase the fragmentations of main memory.

If processes are in fact permitted to process according to process priority, the "recent use" algorithm will tend to best support highest priority processes in respect to code and data segments, and cached domains. If, however, due to a system (or jobclass) "bottleneck" at a shared resource (other than the cpu) processes that tend to take "best advantage" of that resource will be best supported by this algorithm in respect to code and data segments, and cached domains. It is under these circumstances that the memory management algorithm may dramatically fail to achieve the intended results, and ALLOW VERY LITTLE MANAGEMENT CONTROL OVER PRIORITY OF PROCESSING! Very I/O intensive systems with software disc cacheing, particularly those with multiple job classes, could find that the "recent use" algorithm causes the overall system to perform very poorly.

Aside from this general overview I do not intend to pursue this subject any further, except as it relates to disc cacheing. Memory management, disc cacheing, and shared buffering share an interesting characteristic, namely that input and output rates are dependent upon the number of processes. Before concluding this paper, we will look at a very interesting problem that can develop from this characteristic, and may have serious overall consequences on the performance of disc cacheing and cached systems.

DISC I/O, SERIALY REUSEABLE and CONCURRENT

The characteristics of disc I/O are dependent both upon the hardware and the software. So we'll quickly summarize both.

Hardware

Disc I/O, on the HP3000, can have the characteristics of concurrency and serially reuseable, depending upon the configuration. In all cases, all disc I/O's to one specific drive are always serially reuseable. In general, an I/O request and transfer must travel through a channel and a controller --only one device on a channel or controller may be active at

one time. So we may classify all disc drives on one channel as serially reusable resources. If, however, disc drives have separate channels and controllers, the disc drives may operate concurrently.

Manufacturers of disc drives generally publish the characteristics of the disc drives that they manufacture. Of interest to us are those related to performance, namely:

- 1) Seek Time: the time required to position the access arm, containing the heads, at the cylinder with the specified record. This is generally an average over the whole disc.
- 2) Rotational Delay: The time required for the requested record to arrive under the read/write heads. This is generally published as the time for one-half rotation.
- 3) Transfer Time: The time required to transfer a given block of data, after the seek and rotational delay.
- 4) Controller Overhead: The time the drive spends on behalf of an I/O operation not due to any of the three preceding items.

For the HP793X drives HP has published the following characteristics [HP]:

| Function | Average Time |
|-------------------------|--------------|
| Seek | 24.0 ms |
| Rotational Delay | 11.1 ms |
| Transfer time (1 kbyte) | 1.0 ms |
| Controller Overhead | 3.5 ms |
| Total Average Time | 39.6 ms |

As can thus be determined from the above averages, the maximum number of disc I/O's per second to a 793X drive of truly random requests is about 25 (not considering the cost of system overhead). Although we have been frequently told that we may expect as many as 30 I/O's per second from this drive, on a "busy" machine with a high level of multiprogramming it is far more likely that we will average closer to the 25 I/O's due to the greater "randomness" induced by I/O requests being interleaved among processes accessing many different files.

Channel Latency

Frequently rotational delay is referred to as "latency". I avoided using this term as improvements to disc drive technology might make use of this term somewhat ambiguous. Sometimes our biggest concern is the "channel latency", the time a channel is held by a disc drive. Under

past technology channel latency was roughly equivalent to rotational delay (actually it was equal to rotational delay plus transfer time). Newer technology has made it likely that channel latency (on behalf of one drive) has no correlation to rotational delay.

When multiple disc drives are on the same controller or on the same channel it is obviously necessary to prevent simultaneous transfer of data from/to several drives. For this reason it is necessary to employ some kind of "lockout" strategy on any shared channels and or controllers. With past technology a drive would perform a seek if necessary, obtain the channel and controller, await the data to arrive under the read/write heads (rotational delay), proceed with the data transfer, and finally release the channel and controller. That is in part why rotational delay is an important characteristic of a given disc drive. While seek time is far larger than rotational delay, it is only a factor to those disc I/O's on that particular drive. However, rotational delay (and transfer time) affects any devices (usually only discs) on the same controller and/or the same channel.

More recent technology has attempted to reduce this channel latency most notably by "rotational positional sensing" (RPS).

Rotational Positional Sensing and Buffer Prefill

The HP793X disc drives have a feature called RPS for rotational positional sensing. The primary goal of this feature is to reduce the channel latency by waiting as long as possible in the disc platter rotation before trying to obtain exclusive access to the channel (this requires some intelligence on the part of the disc drive and of course would contribute to "controller overhead"). In theory this strategy should be able to come close to reducing the channel latency to the transfer time. In practice, this strategy has contributed to very substantial reductions in contention between disc drives for a controller or channel by substantially reducing the likelihood that the controller or channel is busy. The actual benefit, of course, is very environment dependent.

One of the problems with RPS is that if the channel or controller is busy when the drive tries to obtain it, the drive will not be ready to try again until another full rotation of the disc platter has occurred. With enough activity from several disc drives on one controller or channel it is very possible that RPS WILL DEGRADE PERFORMANCE! Without RPS a drive may be able to try several times to obtain the channel and controller before the data is under the read/write heads; without RPS since it has waited until the last possible moment it must wait an additional rotation before it can try again. The additional rotation that an RPS drive must wait is a full rotation which of course is double the average rotational delay. So just one miss at least triples the average rotational delay. In short RPS reduces channel/controller contention thereby increasing throughput on the channel/controller, RPS, alone, is most effective when used on very busy channels/controllers

with multiple disc drives that are not approaching their point of saturation.

In order to reduce the potential for degradation with RPS, HP has also implemented an internal buffer on the 793X drives. When a disc transfer is 4096 bytes or less the data can be transferred into this internal buffer which will eliminate the need to wait another rotation if the channel/controller is busy. The drive may attempt to obtain the channel/controller more frequently. It may be worth keeping in mind here that large transfer requests either due to large file block sizes, large multi-record transfers, and/or large fetch quantum set with software disc cacheing that exceed the internal buffer size are transferred without use of this buffer. A very busy system with such large transfers may (or may not) benefit from disabling RPS if the disc drives are approaching their saturation point.

Controller Cache

One of the latest enhancements available to 793X disc drive is that of "controller cache". The controller cache is a megabyte of memory used for "buffering" disc transfers. The intention of the controller cache is to reduce the number of seeks and rotational delays for process' read requests by transferring pages (4096 bytes) of requested disc regions to memory contained in the disc drive. Each time a block of data less than or equal to 4096 bytes is requested from the drive it will search its pages of cache for the requested block. The value of this strategy is very dependent upon the nature of I/O transfer requests, but is very likely to improve performance in most environments that are not currently benefiting from disc cacheing.

Software

Since MPE-IV, the HP3000 operating system has begun to use an I/O prioritizing scheme. In general when a process posts an I/O request it is given a priority equivalent to its process' priority, which of course should favor the I/O's of higher priority processes. Any resource other than the CPU, which happens to be the limiting system (jobclass) resource and makes adjustment to process priorities (SIRS, and IMAGE data bases) MAY EFFECTIVELY ELIMINATE ANY BENEFIT FROM THIS I/O PRIORITIZING SCHEME! For example, a low priority job accesses a data base heavily used by high priority processes, in all likelihood each time the low priority job obtains the data base a high priority process will attempt to obtain it, be impeded, and temporarily raise the priority of the job to the impeded process' (high) priority, ensuring that any disc I/O's posted on behalf of the job are placed at the higher priority. (See the "Temporary Reprioritization" paragraph in the CPU section).

Although typically small in comparison to the service times due to hardware, the operating system consumes some overhead in setting up the I/O (which usually requires process dispatching also, when the current process is blocked) as well as processing the interrupt that occurs when

the I/O has completed (which also may require process dispatching, to reawaken a blocked process).

DISC I/O and the FILE SYSTEM

The MPE file system is a process' interface to performing input, output, and control operations to peripheral devices. We will not concern ourselves here with devices other than discs, but much of what is contained here could apply equally as well to other devices.

Now that we have some knowledge of the characteristics of the 793X disc drives we can examine various ways in which we might affect the performance of disc I/O. From what we have seen we can conclude that it is desirable to maximize the likelihood that sequential data transfers reside at physically "near" locations on the disc drive, achieve a good balance (spread) of the total I/O across available drives, and/or reduce the number of data transfers.

Maximizing Contiguously Located Data Transfers

The MPE file system allows disc operations only on files. A file is composed of from 1 to 32 contiguous areas of disc space called extents. The maximum number of extents that a file is permitted is determined (specified) at file creation time.

The fewer extents a file has, the more likely that the data will be located more contiguously on the disc platter (with large extents, however, it may be much more difficult, or impossible, to find enough contiguous space). Assuming no competition with other I/O requests to a particular drive, a sequential read of a one extent file should require fewer "seeks" than a sequential read of a 32 extent file. But this is not as important as we may have heretofore been led to believe because at worst it will save 31 "seeks", and of course with a "busy system" and "high level of multiprogramming", the likelihood of exclusive access to the particular drive becomes very remote.

Achieving Balance of I/O Across Disc Drives

Much has been said about the value of spreading (balancing) disc I/O requirements across disc drives, and there are two very good reasons for doing this: 1) further concurrency (parallelism) of various I/O, and 2) reduction of competing and interleaving I/O seeks.

Further Concurrency of I/O

It is very obvious that when there are several outstanding I/O requests that those which happen to be on different drives can be processed in parallel (concurrent). What is not so obvious is that if there exists a bottleneck in the path to (and including) a particular disc drive (or set of disc drives), no amount of effort spent trying to move files

around will result in improvement in performance to the processes affected by that bottleneck (unless of course, the other bottleneck is somehow related to activity on the disc drive).

Reduction of Disc Seek Time

A good balance of I/O spread across available drives can also help to reduce seek time. For example, if several processes are performing serial reads concurrently on one particular drive, their requests will tend to interleave each other increasing the likelihood of a seek being required upon each access.

On a high volume system, no amount of file movement is likely to prevent these interleaved seeks (although we may be able to reduce them); on a system that is highly variable in demand, such efforts will likely result in larger variations in system performance.

Reduction of Disc Accesses

It is also quite obvious that reduction of disc accesses will provide a corresponding improvement in I/O performance. Unfortunately the most obvious way of accomplishing that is to stop processing as much data, but this solution does not seem to be very popular.

Fortunately there are several more popular ways of reducing disc accesses, one of which is relatively easy to implement, and several others somewhat more difficult to implement (and somewhat more dubious in value).

Buffered Disc Access

Under default conditions a process' access to files are buffered by the file system, which simply means that the file system utilizes an area of memory (frequently an extra data segment, but sometimes an appendage of the process' stack) to stage record transfer to and/or from the disc drive.

With buffered access to a file, the file system transfers data one block (physical record) at a time between the disc and the file system buffer. Each extent of a file is composed of one or more blocks (physical records), and each block is composed of one or more logical records (set of data items). At file creation time the logical record size and block size (actually the blocking factor) are set (specified). Using buffered access, the file system will handle blocking and deblocking of logical to physical records for the requesting process.

Frequently to the detriment of performance, the default blocking factor (logical records to physical records) assigned by the file system at file creation time can be very inefficient both in terms of I/O performance and disc space usage. Since buffered file access is performed on block at a time, the specification of blocking factor can have a very

large impact on performance as it will determine the number of disc accesses. In fact with a sequential read, simply doubling the blocking factor can be expected to reduce disc I/O wait times by about half (assuming no large queueing delays). Since disc I/O wait times are frequently the largest component of processing time, APPROPRIATE BLOCKING FACTORS MAY CONTRIBUTE MORE TO IMPROVING PERFORMANCE THAN ANY OTHER FACTOR.

So shall we proceed to double all blocking factors? As you might have expected, there are some consequences to be considered. Doubling of block sizes will also double the requirement for file system buffers. If this is going to pose a dramatic impact on memory requirements then a compromise may be in order, or we may wish to evaluate the number of buffers assigned to the open files.

Choosing the Number of Buffers

At file open time the file system allows a process to specify the number of file system buffers that will be used during the course of that processes' activity against the file. No discussion of blocking factors should be complete without some discussion of a closely associated factor, namely the number of buffers to be used.

Since the file system cannot possibly predict the pattern of access to a file that a process will be using, by default (with a few exceptions) the file system will assign two buffers for each file open by a unique process. In general this is likely to provide good performance for sequential access to files.

With buffered access, the file system will attempt to minimize the disc I/O delay by performing "anticipatory reads" and "no-wait" writes to a file. Every attempt will be made by the file system to keep buffers filled with "fresh" data. The default case of two buffers allows one to be processed while the other is either being filled or emptied. This works very well for sequential access, as no special knowledge is required by the file system to determine the next record to access, or whether or not a process is finished processing a block.

In many types of "direct" or "random" access to a file, however, the efficiency of the file system's strategy to reduce I/O waits is at best reduced because it cannot possibly know the future pattern of access to the file. In such cases, relying upon the default number of buffers assigned by the file system may not be the best approach. For truly random or largely variable access to a file, an increase in the number of buffers is probably in order. The larger the number of buffers, the greater the likelihood that a requested record may already be in a file system buffer (a "buffer hit"). The more frequent "buffer hits" occur, the fewer disc accesses required.

(The file system provides the "FREADSEEK" intrinsic to allow random accessors, using buffered access, to perform their own "anticipatory reads").

As a general rule we may decide that files that are sequentially accessed should have large blocking factors and few buffers (but not less than two), and files that are to be "randomly" accessed should have smaller blocking factors but many buffers.

Frequently there is a special case to consider relating to temporary scratch files. In order to circumvent the limitations of the small stack size on the "pre-spectrum" HP3000's, or to allow for processing of infinite (or more properly indefinite but potentially very large amounts of information), it is often a good strategy to use a scratch file to temporarily stage this information. Often it is possible to specify a block size (blocking factor) large enough such that no disc accesses will be required (except for initialization and swapping) in the usual case no matter how many reads or writes the process is performing against the file. In this case, for constant access to the same block more than one buffer is not helpful and will only waste memory.

Sharing of Buffers

Contrary to popular belief simple shared access to a file does not imply the use of shared buffers. When using file system buffering the only time buffers are shared is when they are opened with "multi" or "gmulti" access (message files and spooled device files always use "gmulti" access and "multi" access respectively).

Using shared buffers can further improve performance. In the sequential access case, by using shared buffers, we can frequently greatly increase the blocking factors without increasing (in fact usually decreasing) the memory required by file system buffers (of course with more than one accessor it is likely to be desirable to request more than the default of two buffers). In the random access case, shared buffers can allow one to increase the number of buffers that one process may access without increasing (or again frequently decreasing) the memory requirements, and also increase the number of "buffer hits", as each process' view of the buffers will contain the other processes' view as well.

Now before you rush off to start changing file commands or file opens to force the sharing of file system buffers, be aware of several things. When any updating is being done to a file with shared buffering, inadequate locking strategies are far more likely to cause problems than with the non-shared buffer case (even though the locking is just as inadequate in either case). Also, shared buffer access causes the file system to treat the file as a "multi-access" file, and as such requiring a special table entry. Up until recent operating system releases there may have been a very finite limit (if I recall correctly it was 50 at one time) to the number of "multi-access" files that could be opened; and worse yet, at one time even when such files were closed, "room" was

not made for other "multi-access" files (I assume this is not currently the case as \$stdin and \$stdlist, when spooled, are also "multi-access" disc files).

Unbuffered Disc Access

In addition to the default method of buffered access to files, the file system allows "unbuffered" access to disc files. What this means is that all transfer will be made by the file system directly between the process specified area and the disc. There are several reasons why one might choose to use unbuffered access, and (as you no doubt have come to expect, there are implications associated with each of them).

The first reason one might choose to request unbuffered access might be to eliminate the additional overhead inherent in the intermediate staging of the data. This additional overhead always includes the additional cpu time for the extra memory to memory moves. If the file system buffer is not in the same data segment as the process' stack (which is frequently the case), the extra data segment that contains the file system buffers may be temporarily absent from memory (which will require an additional I/O to retrieve it) before any logical or physical I/O can occur.

Many past discussions on this topic have concluded that unbuffered access to a file is always superior, but this is not so! Without building a large amount of "scaffolding" to implement it, unbuffered file access does not permit sharing of buffers between processes. While the argument regarding swapping of extra data segments containing file system buffers is not untrue, it doesn't tell the whole story either. When the file system chooses to use an extra data segment for file system buffers, it will attempt to fill any and all of these data segments with file control blocks for this and other processes' files. Where there is a large amount of control blocks for various files, there will be more likelihood that this data segment is present in memory (even more likely than a particular process' stack) due to the amount of sharing. And a (potentially serious) problem with unbuffered access is that, when a physical disc I/O is posted the source or target data segment is "frozen" in memory until the transfer is complete. Data segments which are "frozen" in memory create two types of problems for memory management. A "frozen" data segment cannot be "swapped" out of memory if the space is needed for something of higher priority, and a "frozen" data segment cannot be moved by the memory manager during the process of local or global "garbage collection" (housekeeping or cleanup). In general it is preferable that data segments which are "frozen" be those that are shared (as it is reasonable that this data segment will shortly be needed by another process), and a shared data segment containing file system buffers can reduce the overall number of data segments that need to be "frozen" (as there could be many I/O's pending against the one data segment).



When looking at buffered access to files previously, we discussed the "anticipatory reads" and "no-wait writes" performed by the file system. With unbuffered access to files, the file system does not perform them on the process behalf unless specifically requested to do so. In order to specifically request "no-wait" I/O when using unbuffered access, a process must be privileged, and it would be necessary to set up its own "anticipatory reads".

Unbuffered access allows a process to request transfers to and from the disc in sizes not necessarily equal to the block size of the file (multi-record I/O). This feature has been indicated, in the past, to always be an advantage when used to transfer multiple blocks per request. This statement, likewise, while not untrue, also does not tell the whole story. With respect to I/O performance what matters most is the number of transfers (which of course depends on the size). It is far easier to reduce the number of transfers by adjusting the blocking factor than setting up the appropriate "scaffolding" to handle mutiple block transfers and the associated blocking and deblocking of records. (A very valid use of mutiple block transfers might be the way it allows for varying transfer sizes, i.e., on line processing accesses the file randomly, so block transfers should probably be small, but batch processing accesses the file sequentially, thus transfers on its behalf should be large). We could set small blocks and use buffered access for interactive use, and set up batch processing against the file to use "multi-record I/O" to achieve the benefits of larger block sizes.

One additional point to make regarding 'multi-record' I/O, is that its full advantage is not achieved unless the block size of the file is equal to or an even multiple of the size of one disc sector. Any time "multi-record" I/O requests are made against a disc file whose blocks do not begin and end on sector boundaries, the actual number of transfers will not be less than the number that would occur with single block transfers (each of the tranfers set up by the file system in this case would be quickly one right after the other, so there is not a large window for other requests to interleave this one). However, if the blocksize is an even multiple of the sector size of the disc, the file system is able to retrieve multiple blocks with only one actual transfer.

(The latest versions of MPE-V allow for buffers up to 32k in size, previously the limit was 14k. At the 14k limit, multi-record I/O could be utilized to advantage by allowing one to exceed the 14k limit. But with the new 32k limit, it is more likely that a file system buffer would have room for more data than would be left within a process' own stack).

SOFTWARE DISC CACHING, a SHARED RESOURCE

MPE Disc Caching is an optional feature available on some of the HP3000 systems to help to improve the disc I/O performance of the system.

I will not provide much detail here on the operation of disc caching as there is much in the literature about it. For more information I might suggest [CARROLL] as a good reference.

In very general terms, disc caching's goal is to reduce the "disc I/O waits" that processes would otherwise encounter by:

- 1) transferring larger amounts of data from the disc drives into available main memory, to reduce the number of transfers;
- 2) increasing the likelihood that frequently accessed files, or portions of files, will generally be available in main memory; and
- 3) minimizing the likelihood that processes be forced to wait on a disc write by allowing "posts" to occur and be processed at a "background priority".

In very simple terms, disc caching, at its best, tries to make all disc I/O act as if it were performed with "multi-record", "no-wait", and "anticipatory" I/O's. In fact, a system that is enjoying a large amount of benefit from disc caching is not likely to benefit much from explicit use of "multi-record", "no-wait", and/or "anticipatory I/O's" (except in terms of CPU time). This does not mean to suggest that these features should no longer be used, as explicit use of these features will allow sustaining far more growth on the system than could be achieved with caching alone (as currently implemented).

Costs of Disc Caching

Disc caching uses available regions of main memory, which are handled the same as code or data segments by the memory manager. So disc caching works in competition for other main memory needs, as well as increasing the workload of the "memory manager".

Besides the increased work of the memory manager, additional effort must be spent for each file system I/O request to search for the requested data in the existing cache domains, as well as additional cpu time in memory-to-memory moves from cache domains to file system (or user "no-buf") buffers.

Additional Caching Tips

In the literature that I have seen to date, a very important consequence of disc caching's implementation is, at best, not clear; and since it has very important implications we will take the time to emphasize it here.

Under the usual circumstances cache domains created on behalf of a read request will be far larger than those of a write request (the "fetch

quantums" help determine the "read domain" size, and in general the block size of a file will usually determine the size of a write domain to be posted upon behalf of a process). Caching, however, makes no distinction between "read domains" and "write domains" (as I have done here).

Since caching will "map" any blocks to be posted to an existing domain that has the proper "disc image" (not to be confused with HP's use of the term "mapped domain"), and "flag" that domain as "dirty"; any subsequent records that are to be posted to that same domain will be "blocked" ("write hit") until the previous posting(s) has completed.

The consequence of this is that any process which performs reads before writes (e.g. IMAGE) is very likely to be subject to a large proportion of "write hits".

IMAGE, a SERIALY REUSEABLE RESOURCE

All IMAGE data bases prior to TURBO-IMAGE are serially reuseable resources, that is there is no concurrency permitted between processes accessing a particular IMAGE data base. This characteristic of IMAGE data bases has been frequently been referred to as "single threading".

With TURBO-IMAGE comes some possibility of concurrency. Although, at the time of this writing, I have not yet had the opportunity to work with TURBO-IMAGE, as I understand it TURBO-IMAGE will allow concurrency only during serial reads and non-structural updates. Any other type of read, write, or delete will not permit concurrency. In general I expect this to be a very small window for concurrency during heavy interactive data entry. We certainly could envision some environments that could see a large improvement, however.

An IMAGE data base is composed of a set of logically related files. As IMAGE is implemented, it is actually a layer of structure and protocol imposed upon the file system. Since a large part of IMAGE processing relates to file I/O we should examine the details of its operation.

In order to limit the length of this paper, we will be presuming a certain level of background in the internals of the IMAGE data base management system. If this is not the case, or possibly for purposes of review before proceeding, I would suggest reading the IMAGE/3000 HANDBOOK [GREEN] which is an excellent reference source for details of HP3000 IMAGE data bases.

In addition to the "single-threading" of IMAGE data bases, there are several other characteristics, most notably master "synonyms" and detail "sort fields" and "delete chains" which can substantially affect performance.

IMAGE and the File System

As I indicated, IMAGE may be thought of as an imposition of structure and protocol on the existing file system. From what we have seen about the file system we can immediately see several points worth noting where IMAGE performance cannot quite achieve the performance that could be attained with standard file access.

IMAGE accesses the data set files in an unbuffered manner (from the point of view of the file system). From our knowledge of the file system we know that we forgo the file system's ability to perform "anticipatory reads" and "no-wait" writes (TURBO-IMAGE will allow "no-wait" writes when a "deferred output" option is selected). There is currently no way of implementing "anticipatory reads" within IMAGE.

Additionally (with IMAGE versions prior to TURBO-IMAGE) the number of and size of buffers can be very limiting. Pre-TURBO implementations contain all information on data set and item names, security, locking and buffers in one extra data segment. Since this is all placed into an extra data segment no larger than 32k, we may be very "buffer constrained". TURBO-IMAGE places the buffers into a separate data segment, so this alleviates the buffer constraint somewhat (it should be possible, and helpful, under TURBO, to either increase the size of IMAGE buffers, the number of IMAGE buffers, or both). This single enhancement of TURBO-IMAGE, while not especially emphasized to date, is probably the MOST IMPORTANT performance enhancement (and possibly the only) that large and heavy IMAGE processing shops will enjoy.

IMAGE and Software Disc Caching

Unfortunately IMAGE and disc caching are frequently not an optimal fit for each other. Depending upon the setting of the random fetch quantum (default is 16 sectors or 4096 bytes) and the block size of the data set, we can expect that since IMAGE always performs a read of a block before a write to that block, that a large fetch quantum will favor a large number of "read hits" (good) but also favor a large number of "write hits" (bad). If we try to improve the "write hit" ratio (reduce it) by reducing the fetch quantum, we will be making a corresponding decrease in the "read hit" ratio. Since different data sets and different data bases will have different characteristics of access, it is very difficult (maybe impossible) to know exactly how to achieve optimum read and writes (certainly to attempt to properly do so would require a very large amount of data collection and analysis).

Additionally, the user label which is used to hold control information in each data set, is very likely in a busy data set to continually cause a "write hit" as each "put" and "delete" for all users are continually re-posting it.

Regardless of the mode of read access to a data set, IMAGE performs a file system "direct read" of the file; the consequences of this are that

even in sequential reads of large data sets the caching "random fetch quantum" will be used (as opposed to the caching "sequential fetch quantum").

And finally, IMAGE (as does KSAM) uses the serial write queue (if no deferred output option is selected). This can be a very serious consequence, as its effect is to serialize all writes to any and all IMAGE data bases. In other words, under standard shared access environments while disc caching is in effect, ALL writes to ALL data bases are "single threaded". If using disc caching, any efforts to reduce the single threading by "splitting" the number of data bases, is at best reduced in write intensive environments.

Synonyms

Assuming a good "hashing algorithm" and/or a valid random distribution of search item values in a master data set, there is a relationship between the ratio of free space in the data set to the ratio of synonyms to be expected ([VOLOKH a]):

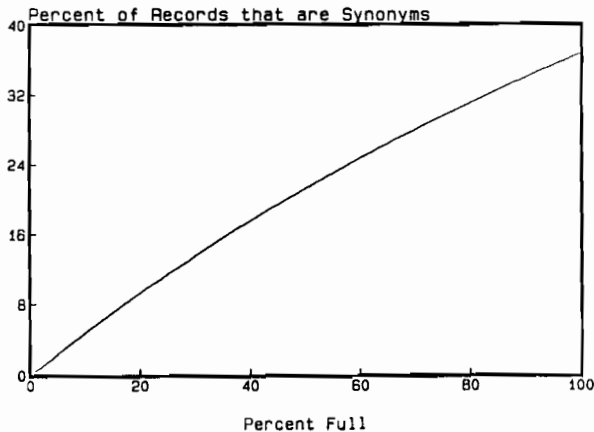
$$S = 100 * \left(1 - \frac{1 - e^{-F}}{F} \right)$$

Where S is the percentage of synonyms

F is the ration of the number of entries to the capacity

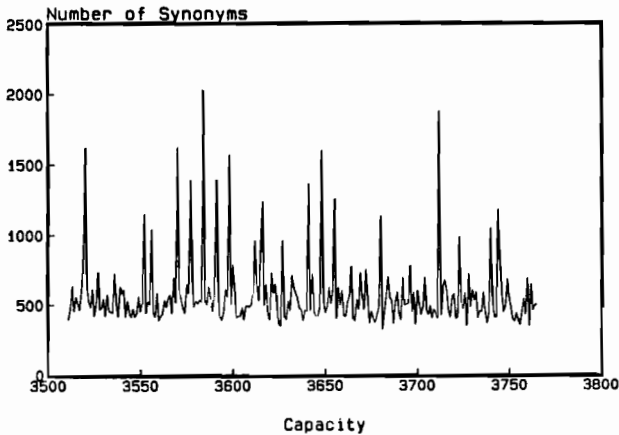
e is the base of the natural logarithm

NUMBER of SYNONYMS
as a Function of Data Set Percent Filled



The primary effect of synonyms that is of concern to us is the effect that synonyms have on I/O performance. In fact there are two ways in which synonyms impact I/O performance: 1) when performing a calculated read by search item value (e.g. DBGET mode 7) a synonym chain may need to be traversed to find the appropriate entry; and 2) when adding a new entry (i.e. DBPUT) a synonym may well require one or more "probes" (forward sequential reads) to find an empty record to use.

EFFECT of CAPACITY on NUMBER of SYNONYMS
For CONTROL-MASTER of MANUF.DATA.SYSTEMS

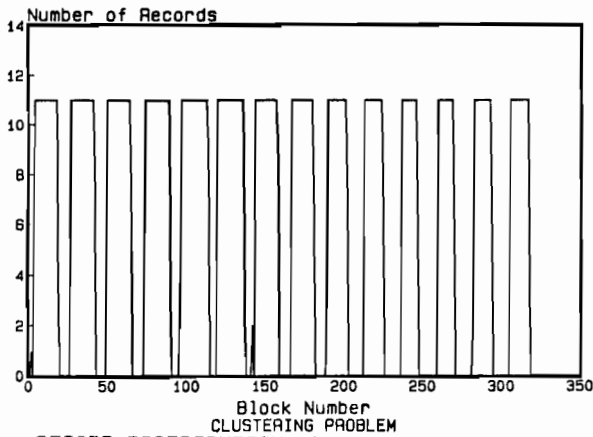


This figure is a plot of the total number of synonyms in a given data set, with a character type search item value, for various capacities (the capacity, of course, being a parameter to the "hashing algorithm"). We have encountered quite a bit of discussion over the last several years regarding the choice of capacity for reducing master data set synonyms. Suggestions have been made that prime numbers do not ensure the best distribution of synonyms. By trial and error I'm sure that you will find that there are other capacities (non-prime) which improve synonym distribution, however, there is no known sequence which can be guaranteed to work as well as prime numbers typically do. Therefore, at present, if one is not satisfied to use prime capacities, trial and error will be the only way of choosing an acceptable alternate.

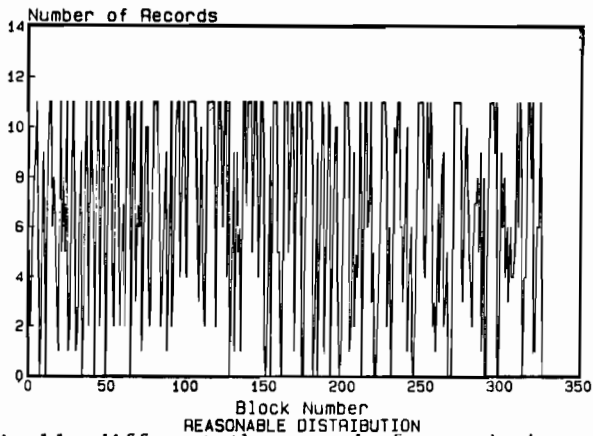
Clustering

It appears, in practice, that "clustering" of (contiguously located) records seems to be far more dependent upon capacity than the absolute number of synonyms. Clustering problems can be expected to have a much larger impact on performance than synonyms alone. The following figures represent a pictorial view of an actual data set, with actual data, and the clustering for two selected capacities:

RECORD DISTRIBUTION at a CAPACITY of 3584
For CONTROL-MSTR of MANUF.DATA.SYSTEMS



RECORD DISTRIBUTION at a CAPACITY of 3594
For CONTROL-MSTR of MANUF.DATA.SYSTEMS



Note how radically different the spread of records is, even though the tested capacities are very near in value. The first one represents a very serious clustering problem, which will have a very dramatic impact on the addition or deletion of records as many physical I/O's will frequently be required to find an empty location for a synonym or to "migrate" an existing synonym.

One of the frequent "rules of thumb" widely circulated, is that a master data set should not be permitted to exceed 80% of its capacity. The object of this rule is to minimize synonyms. As we have indicated, by far the most significant problem relating to synonyms is "clustering" or

contiguously filled blocks. For this reason wouldn't the rule be better replaced by a rule that suggested we allow the ratio of free space to be proportional to the reciprocal of the blocking factor? If this rule were followed we could (assuming a good spread of records) always have one empty record per block. For example, with a blocking factor of 7 we might want 1/7th of the records to be empty, or with a blocking factor of two our goal might be to leave half of the records empty. With this rule in place, the cost of logical "probes" would be very negligible.

Hashed Access

All of the IMAGE intrinsic that access or manipulate records in a master data set must be capable of performing a calculated read, which is performed by "hashing" the value of the given search item to an address and comparing it to the search item value in the record at that address. If a record exists at that location but its search item value does not match that of the given search item then the synonym chain (if any) must be traversed in like manner until either the record is found or the end of the synonym chain is encountered. There are two possible circumstances that may be encountered with respect to a calculated read with synonym entries: 1) the whole synonym chain must be "traversed" (read) in order to determine that the given search item value definitely does not exist; or 2) on average half of the synonym chain must be traversed (read) in order to determine that a given search item value definitely does exist.

Again assuming either good "hashing" and/or a valid random distribution of search item values in a data set, from the ratio of free entries in the data set we can expect the logical reads for an "unsuccessful" find of a given search item value to be [KNUTH a]:

$$LR = (e)^{-F} + F$$

Where LR is the number of logical reads expected

F is the ratio of the number of entries to the capacity

e is the base of the natural logarithm

Further assuming good "hashing" and/or a valid random distribution of search item values in a data set, the ratio of free entries in the data set can also be used to predict the number of logical reads for a "successful" search of a given search item value to be [KNUTH b] and [VOLOKH b]:

$$LR = 1 + \frac{F}{2}$$

Where LR is the number of logical reads expected, and

F is the ratio of the number of entries to the capacity

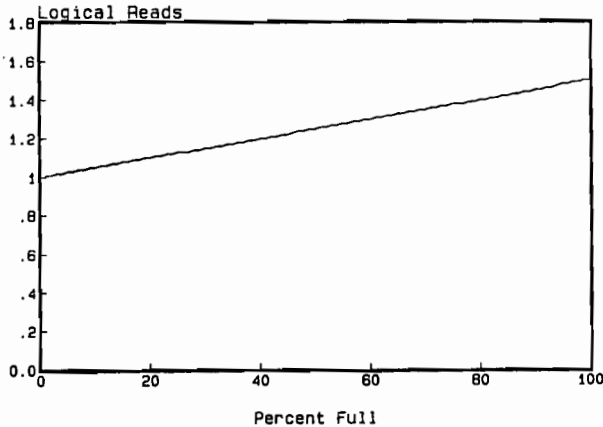
Of far more potential consequence than the logical reads required in a successful or unsuccessful traversal of a synonym chain, is the number of logical "probes" (records read while looking for an empty entry) necessary to locate a free entry when trying to add a new synonym entry or "migrate" an existing synonym. As you might expect, we can also predict this¹ making the same random assumptions [KNUTH c]:

$$LP = \frac{(C + 1)}{(C - E + 1)}$$

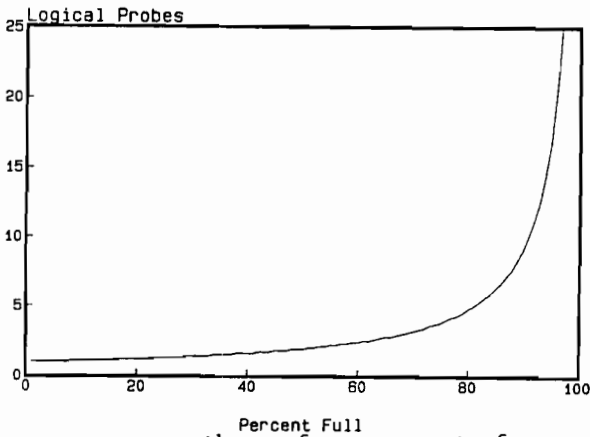
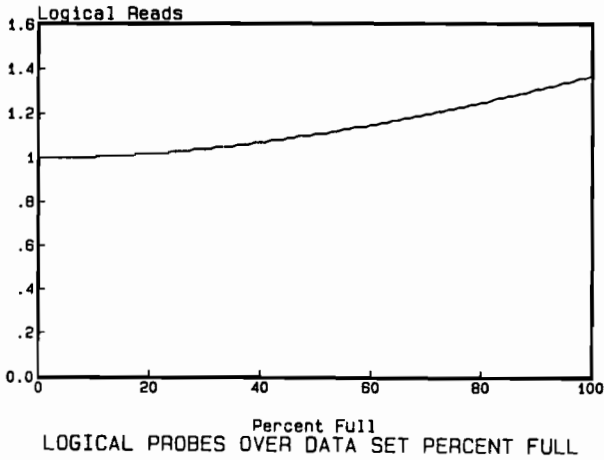
Where LP is the expected number of logical probes
 C is the capacity of the master data set, and
 E is the number of entries in the data set

The following figures represent the expected logical I/O's (reads or "probes") to handle the successful search, unsuccessful search, and "probes" for synonym entry or migration:

LOGICAL READS for a SUCCESSFUL ACCESS
 Over Data Set Percent Full



LOGICAL READS for an UNSUCCESSFUL ACCESS
Over Data Set Percent Full



Once again, as you can see, the performance cost of synonyms is largest in the logical "probes" case (related to clustering). For randomly distributed data sets the successful and unsuccessful "logical reads" can be expected to be a very negligible cost.

A "DBFIND" or "DBGET mode 7" must perform the logical read steps (which will either be successful or unsuccessful). A "DBPUT" must start by performing the logical read steps (which could be successful or unsuccessful) and if unsuccessful will need to perform the logical "probes" steps each time the new entry is a synonym to an existing entry, or a

primary to another entry exists at the primary hashed address for the new search item value.

Detail Data Sets

As we know (or are assuming we know) the use of a sort field on a detail chain can impact the I/O performance of a detail data set. We will not go into much depth here (again I'll refer you to the IMAGE 3000 HANDBOOK [GREEN a]). If a sort field is used on a detail chain we might look at the results in three ways: 1) the records are added in already sorted order, 2) the records are added in a non-sorted and truly random order, and 3) the records are added in reverse sorted order.

If records are added in already sorted order no additional logical (or physical) I/O's are ever required to add the entry. If records are added in a truly random order, each additional entry will require the additional physical reads of, on average, half of the chain length existing at the time of each entry. If records are added in reverse sorted order, each addition will require physical reads of all records on the chain existing at the time of entry for each entry.

The "delete chain" maintained in a detail data set when entries are deleted can also impact the I/O performance of an IMAGE data base. The ultimate effect of the delete chain is that entries placed into these deleted locations will tend to be widely distributed, increasing the likelihood that each logical I/O will require a physical I/O.

The Data Set Control Record

It seems that very little attention has been paid, in other discussions, to the IMAGE data set control record (located in the "user label" area of the file). There is an important consequence to this record that should not be overlooked, namely that when a data base is accessed in any "non-deferred output" mode, each "PUT" or "DELETE" to the data set will cost an additional physical I/O, just to update the control record. And what may be of even more consequence is that, for large data sets, the fact that this record is constantly being updated along with the standard records, a large data set may always require two (slow) disc "seeks" within the course of one "PUT" or "DELETE" even if no other processing is occurring on the same disc drive.

IMAGE Block Sizes and Buffers

When a data base is created, IMAGE permits some (limited) control over block sizes with the schema "\$Control Blockmax". Maximum block sizes may be set from 128 to 1024 words per block. In general, the same considerations of I/O performance relating to the file system would apply equally as well here, so we will not pursue that aspect again.

As we would expect, typically a large number of buffers (obtainable through small block sizes) favor interactive use in a multi-user

environment, and large blocks favor batch-type processing (where there is little concurrency against the data base). Unfortunately, as with standard file system access, IMAGE is not flexible enough to allow both, so we must either compromise or lean in one direction or the other.

Depending upon the linkages involved, a potentially large number of buffers may be required within the course of a single intrinsic. For example a "worst case" DBPUT to a detail set could require:

$$(4 * \text{auto masters}) + (1 * \text{manual masters}) + (2 * \text{paths}) + 1.$$

The point that this emphasizes is that, particularly during interactive use, we need to ensure a large supply of buffers. A shortage of buffers can cause serious performance degradation. The performance degradation due to a "buffer supply crisis" is very seriously compounded when ILR is used, as it significantly increases the I/O requirements of ILR (only with a "put" or "delete").

Even with the control afforded by IMAGE over the definition of block sizes, IMAGE frequently does a very poor job of taking advantage of the block sizes of which it may be capable for some of the data sets. For this reason, performance of an IMAGE data base can be improved by using up some of this wasted space that IMAGE may leave at the end of a block. This can be accomplished by a modification of the "root file" before DBUTIL is used to create the data base, or optionally can be performed more easily with third party software.

Comments on "Single Threading"

Much has been said about the effect of "single threading" of IMAGE data bases in the past. And much disappointment has been indicated in regard to little alleviation of this problem with TURBO-IMAGE.

We should not ever be alluded into thinking that there can ever be full concurrency when structural elements of a data base are being updated in a shared environment while ensuring integrity. Certainly others have succeeded in increasing the "threads of control" by reducing the "amount of structure" that must be held exclusively.

If the "single threading" were wrapped around maybe a few hundred thousand instructions on a one MIP machine, the consequence would be relatively small. In fact the "single threading" is wrapped around possibly a very large number of disc I/O's, which due to the slow disc I/O transfer rates, can cause very serious performance consequences. If IMAGE were structured to allow concurrent "no-wait" I/O's to be occurring in the course of a "PUT", the time the data base is held exclusively could be reduced to very near the time of two I/O's regardless of the structure involved (assuming data sets were located on different disc drives). This would not only require a change to IMAGE to implement the concurrent I/O's, but could also require a change to the file system to

allow several "no-wait" I/O's against the same file. The greatest benefit of such a strategy would be that the larger systems performance with IMAGE (in terms of I/O) would not differ significantly from the smaller systems.

Predicting the Performance of IMAGE

The formulae we have looked at, are in terms of "logical I/O's". Of concern is the "physical I/O's" actually required. "DBLOADNG" and "HOWMESSY"² can provide us the information necessary to convert the "logical I/O's" to "physical I/O's" by way of the "inefficient pointers" values for each data set.

The first effort at trying to predict and/or improve IMAGE performance should be to ensure that master data sets characteristics are in fact reasonably near the expected characteristics (from the preceding formulae), and if not, to make the appropriate corrective action.

When satisfied with the actual data set characteristics, the formulae we have looked at, adjusted by "inefficient pointers" and/or "block factors", should give us a very reasonable estimate of the number of I/O's required for various IMAGE operations (remember that the I/O's are the greatest elapsed cost of IMAGE, and all shared logical writes are equal to one physical write). At first it might seem that the calculations might become a bit tedious, but I might suggest that an average programmer should be capable of creating a program within an hour to make the appropriate calculations, and which could even determine (by "DBINFO") existing linkages.

OTHER RESOURCES

There are potentially several other resources that may be of concern whether they be hardware, software, or a combination of both. We should be aware that others exist, but we will not concern ourselves with any others at this time.

2. QUEUING THEORY and QUEUING NETWORK ANALYSIS

Queuing theory is a branch of applied mathematics which studies waiting line processes through the use of mathematical and/or simulation models to aid in determining the effect of queues, to help understand fluctuating demand, and potentially enable one to better control provision of services.

Queuing network analysis, on the other hand, is a close relative or subset of queuing theory; its advantage is that it restricts itself to networks of queues such as we find in computer systems, but does not require a detailed understanding of queuing theory to successfully

apply. When the appropriate queuing networks from a computer system are chosen, queuing network analysis allows for surprisingly accurate and efficient evaluation. For more information on the subject of queuing network analysis, I would recommend [LAZOWSKA].

Considerable progress has been made in queuing network analysis in recent years toward a method of developing very accurate performance models of multiprogramming systems. In general a validated model can be very useful in calculating or gaining insight into performance metrics that have not been or cannot be accurately measured. And of course, perhaps more importantly, a validated model can be extremely useful for purposes of prediction. Such models can frequently be capable of predicting throughput and utilization to within 5% of the actual, and mean lengths and response times to within 25% of the actual³.

The mathematics of queuing theory/queuing network analysis are well developed and understood, the difficulty lies in the proper application to the system under study. This may require a very deep appreciation of the internal characteristics of each of the components of the system under study (although frequently many details of the system can be ignored while still successfully creating an accurate model) along with techniques that are best polished through experience.

While it may be unfortunate that such an analysis would far exceed the scope of this paper, that fact will not prevent us from using concepts from the theory to provide some valuable insights into the performance of our collective systems.

Almost regardless of our frame of reference today we find ourselves being concerned with various forms of queues or simply waiting lines. The population explosion has brought us congestion, and congestion brings us queues. As we have all experienced, queues form any time customers arrive at a facility at a rate that is faster than the facility is capable of servicing them. Let's examine a hypothetical one.

Imagine the post office. Let's assume we are observing a local post office over a fifteen minute period. At this particular post office the service time is always exactly one minute. What would happen if the post office was experiencing a "worst case" arrival scenario, namely that all customers were arriving at the very same instant at the beginning of our observation period?

If only one customer arrived at that instant his wait time would be one minute, as he need only to wait the service interval. If two customers arrived at the first instant of our observation, the first would wait one minute, as before, but the second would wait not only one minute for his service but also one minute in line waiting for the first to be serviced; this would place the average wait time for both of them at 1.5 minutes, a 50% increase in average wait time due to just one additional customer.

The following chart shows the average waits for the given number of "worst case" (simultaneous) arrivals at the beginning of the observation period.

| CUSTOMERS | AVERAGE WAIT |
|-----------|--------------|
| 1 | 1.00 |
| 2 | 1.50 |
| 3 | 2.00 |
| 4 | 2.50 |
| 5 | 3.00 |
| 6 | 3.50 |
| 7 | 4.00 |
| 8 | 4.50 |
| 9 | 5.00 |
| 10 | 5.50 |
| 11 | 6.00 |
| 12 | 6.50 |
| 13 | 7.00 |
| 14 | 7.50 |
| 15 | (saturation) |

THE WORST WAIT

As is clear in this "worst case" scenario the average wait time grows steadily larger as the number of customers arriving simultaneously in the observation period grows larger.

On the other hand, though, if we assumed a "best case" arrival rate, namely that each customer arrived after the previous one had completed service (the arrival rate less than or equal to the service rate), there would never be any waiting time in the queue, so the average wait time would be equal to the service time or one minute.

Now that we know the extremes, what can we expect in reality? As we can see, the interarrival times can have a very dramatic effect upon queue lengths and wait times. Intuitively we would expect reality to fall somewhere between the extremes, and that the mean queue length would depend not only upon the arrival to completion rate (arrival rate and service times), but also the distribution of interarrival times. Queuing theory/ queuing network analysis concern themselves with the random probabilities of customers arriving for service and/or varying service times, and are powerful tools to aid in our understanding of the effect of queues.

Understanding the effect of queues is a very important foundation to understanding performance issues, as frequently, response times or turnaround times are more directly dependent upon queue lengths than resource service times. In fact, where a high level of multiprogramming exists, wait times for critical resources are typically more directly queue dependent than resource service time dependent.

In a typical random environment with one or more highly utilized resources, wait times WILL be far more dependent upon queues than the resource's service time!

Now, in our post office example, if we were instead to assume random (poisson) arrival times, queuing theory provides us with the following formula to determine the expected mean number of customers in the system per unit of time:

$$L = \frac{(U)}{(1 - U)}$$

where U is the utilization of the device and
L is the mean number of customers in the system

Queuing network analysis provides us the following formula to determine the wait time (response time) per unit of time:

$$R = \frac{L}{O} - (Z)$$

where R is the mean residence time (frequently considered response time, wait time, or turnaround time)
L is the mean number of customers in the system
O is the mean output rate
Z represents the mean "think" time of interactive processes (zero in this example)

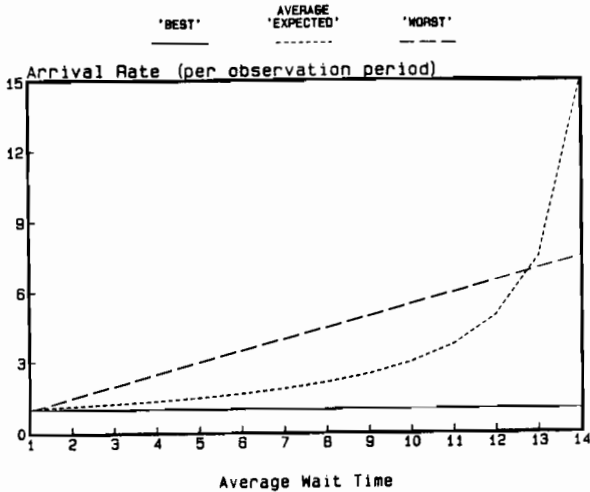
Given the post office scenario and these formulae and our specified time interval, we could then expect the following:

| CUSTOMERS | AVERAGE WAIT |
|-----------|--------------|
| 1 | 1.07 |
| 2 | 1.15 |
| 3 | 1.25 |
| 4 | 1.36 |
| 5 | 1.50 |
| 6 | 1.66 |
| 7 | 1.88 |
| 8 | 2.14 |
| 9 | 2.50 |
| 10 | 3.00 |
| 11 | 3.75 |
| 12 | 5.00 |
| 13 | 7.50 |
| 14 | 15.00 |
| 15 | (saturation) |

EXPECTED WAIT

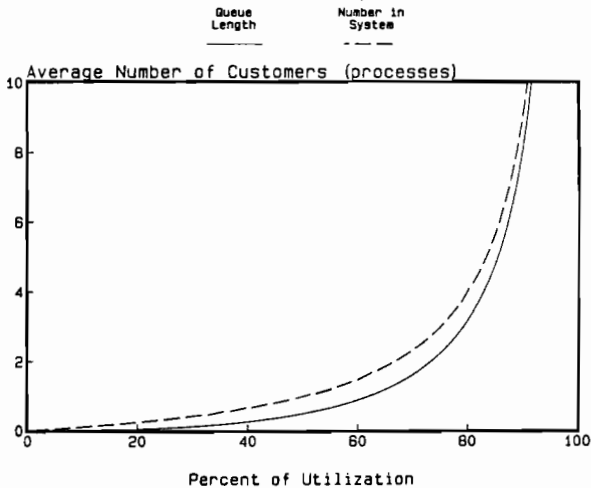
As you can well see, although the 14th customer feels sure that the attendee at the post office is particularly slow today (and is certain to inform his congressman of that fact), the wait of the 14th customer is due almost entirely to the queue!

'POST OFFICE' QUEUEING



Since there is a direct relationship between resource utilization and both the mean queue length of the number of customers (processes) in the system, it is interesting to look at that relationship.

RESOURCE UTILIZATION to QUEUE LENGTHS and Number in System



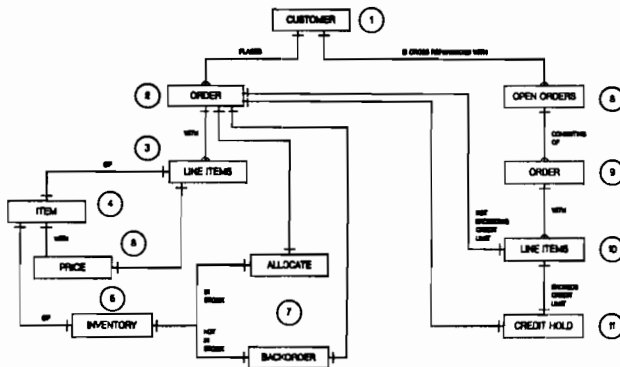
This chart represents the average number of processes in the system (waiting and/or being serviced) and the mean queue length as a function of utilization. Note that the mean queue lengths grow to infinity (yes, the mean lengths not just the absolute lengths). If the system were to operate at this rate the mean response times would grow to infinity as well.

Obviously queue lengths in practice do not approach infinity because they are not permitted to (customers leave, and computer systems have very finite limits to the number of processes). But the graph is still very disappointing. If we desire to limit the mean queue length to 5, utilization cannot exceed 85%. An average queue length of 5 implies that sometimes it may be quite a bit longer, and with the very large slope at that point on the curve, the queue length is very very sensitive to changes in utilization in this range.

An average queue length of 5 at either a slow resource or a resource with long mean service times (e.g. a disc drive or data base) will lead to intolerably long response times. So, for interactive processing, WE CANNOT TOLERATE UTILIZATION RATES EVEN APPROACHING 85%, at these resources, AND ACHIEVE GOOD RESPONSE TIME.

3. An INTERACTIVE APPLICATION UNDER the MICROSCOPE

Let us now take a more specific look at an on-line order entry application which solely uses IMAGE. The order entry application is probably rather standard.



Order Entry
Data Flow Diagram

- 1) when an order is placed a CUSTOMER record is read;
- 2) an ORDER record is created;
- 3) multiple LINE-ITEMS are created;
- 4) each line-item is edited against an ITEM record;
- 5) for each line-item the PRICE is read;
- 6) for each LINE-ITEM an INVENTORY record is read;
- 7) the INVENTORY record and ORDER record is updated;
- 8) for each customer record multiple OPEN ORDERS are read
- 9) for each open orders record an ORDER record is read;
- 10) for each ORDER record multiple LINE-ITEMS are read; and
- 11) if the total of all outstanding orders exceeds the credit limit the ORDER record is updated and a CREDIT HOLD record is added (about 33% of the time).

Given this information, the formulae from the IMAGE discussion, the knowledge that the involved sets are well maintained, and the "inefficient pointers" on structural chains, we can estimate the I/O requirements to enter a typical order. We know that each order has an average of 3.82 items on the order, and that each customer has an average of 17.23 outstanding orders. We will be assuming a "busy data base" (few "buffer hits") and either no disc caching or a system "too busy to benefit significantly from disc caching".

| IMAGE CALL | I/O's per | Times | Total I/O's |
|---------------------------------------|-----------|---------|-------------|
| DBGET 7 from CUSTOMER master | 1.1559 | x 1 | 1.1559 |
| DBPUT to ORDER master | 2.8938 | x 1 | 2.8938 + |
| DBPUTs to LINE-ITEM detail | 6.5459 | x 3.82 | 25.0053 + |
| DBGET from ITEM master | 1.0959 | x 3.82 | 4.1863 |
| DBGET from PRICE master | 1.0282 | x 3.82 | 3.9277 |
| DBGET from INVENTORY master | 1.1308 | x 3.82 | 4.3196 |
| DBUPDATE to INVENTORY master | 1.0000 | x 3.82 | 3.8200 |
| DBUPDATE to ORDER master | 1.0000 | x 1 | 1.0000 |
| DBFIND on OPEN-ORDERS (customer) | 1.1559 | x 1 | 1.1559 |
| DBGET 5 on OPEN-ORDERS detail | 0.0430 | x 17.23 | 0.7408 |
| DBFIND on LINE-ITEMS (order) | 1.0880 | x 17.23 | 18.7462 |
| DBGET 5 on LINE-ITEMS detail | 0.9360 | x 65.81 | 61.5981 |
| DBPUT to CREDIT-HOLD master (30%) | 3.4820 | x 0.33 | 1.1491 + |
| DBUPDATE to ORDER master (30%) | 1.0000 | x 0.33 | 0.3333 |
| + I/O's for ILR (3 per, TURBO 2 per): | | | 15.4500 |
| TOTAL I/O's: | | | 145.4280 |
| TOTAL IMAGE Intrinsic Calls: | | | 124.0300 |

Whew! Until we look at something in this amount of detail, it is very difficult to envision the actual costs associated with an operation. Who would have thought that the simple order entry program was this I/O intensive?

Based upon a "busy system" and "busy data base" (few IMAGE "buffer hits") assumption (and no help from any form of caching) if we are achieving a rate of 25 I/O's per second this average order entry will take about 5.82 seconds (not including I/O's for user logging).

This is already outside of, or at the outer extreme of what is generally considered acceptable response time.

It is not hard to see how such a program can almost single-handedly bring the data base to full "busy data base" form. The number of I/O's it needs in the course of it is probably about ten times the average total number of buffers available.

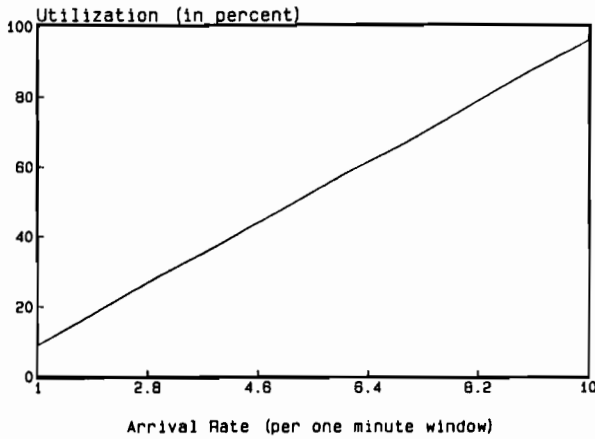
Likewise, the large number of "unrelated" I/O's this program entails could probably involve a substantial percentage of the cache domains on a "busy" system with software disc caching.

Somewhat less obvious, but possibly more important in the end, is the degree of vulnerability that this program has to other processing on the system. We can get an idea of this vulnerability by both the number of I/O's and the number of IMAGE intrinsic calls.

Each time a physical I/O is requested the process is blocked, and if available another process is dispatched. This provides ample opportunity for low-priority "resource hogs" to be dispatched. Likewise each IMAGE intrinsic call creates some vulnerability, as the "single threading" of the data base will cause a "process block" if any other process is using the data base. As such, with both the large number of I/O's and the large number of IMAGE intrinsic calls, this particular process is extremely vulnerable to the load on the system. This extreme vulnerability frequently shows itself as widely varying response times. (Frequently it is this variation, rather than slowness, that is most disturbing to users).

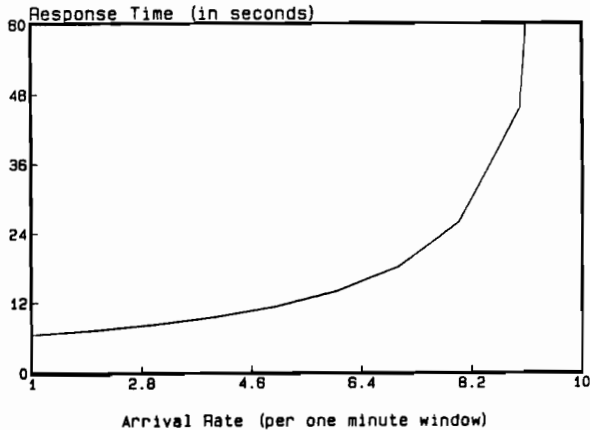
Now that we have collected this information for a typical run of a single order entry process, let us use this information in light of queuing theory to see what response curves would look like for this data base assuming the only activity was order entry⁴.

AVERAGE ARRIVAL RATE to DATA BASE UTILIZATION



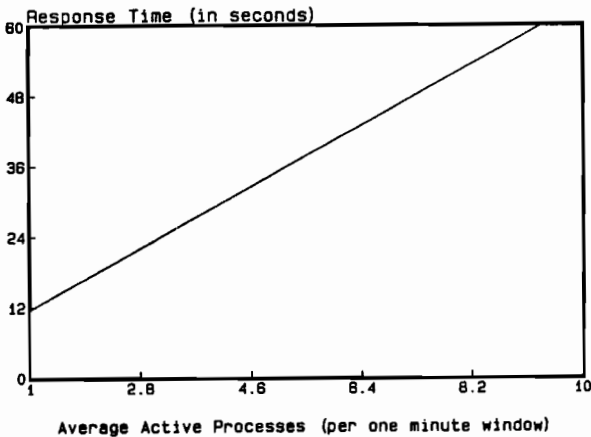
Since we can process slightly more than 10 of our average transactions (or equivalent workloads) within a one minute window, that is the point at which the resource utilization reaches 100%, otherwise known as the point of "saturation". More particularly, saturation of a resource is the point at which queuing absolutely must occur. As we would expect the relationship between average arrivals and average data base utilization is a linear relationship. Unfortunately, it is frequently (and erroneously) assumed that the relationship between average arrivals and response time of interactive applications is a linear relationship.

AVERAGE ARRIVAL RATE to RESPONSE TIME



As you can see, interactive response time grows at an accelerating rate for increasing average interarrival rates. The response time in this scenario rapidly exceeds acceptable limits. The nature of this accelerating relationship is not as important as the relation between the "number of active processes" (i.e., those processes that are either executing, or queued at this resource) and the response time.

ACTIVE PROCESSES to RESPONSE TIME



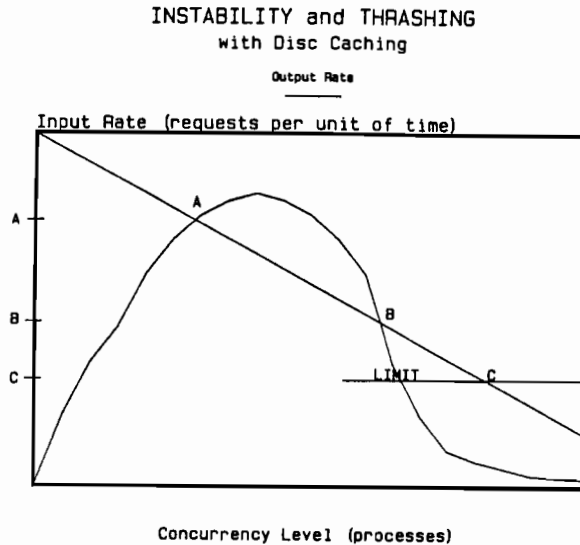
This plot probably paints the most dismal picture of the order entry workload. In order to maintain (on our "busy system" and "busy database") only one process active, on the average, at all times, the response time will average about 12.0 seconds per transaction.

4. CACHING REVISITED

We have been looking at a "busy system" scenario. Part of the "busy system" assumption includes the assumption that if disc caching is available, that the system is so "busy" that the current workload is not benefiting (or is degraded) by it.

I would like to take the opportunity here, as I had promised in the background discussion on memory management, to take a look at an interesting characteristic that is shared by memory management, caching, and buffering (the point here relates to software disc caching). This discussion will hopefully provide some insight into a scenario that causes software disc caching to become very degrading and predatory beginning at a time in which demand for cached I/O is rather heavy, but surprisingly continuing well into a period of modest I/O demand. For an excellent reference on this subject I would suggest [COURTOIS].

It is well agreed that if we were to plot a curve characteristic of the output function of memory, caching, or buffering it would look something like the following:



This curve is characteristic of a resource with an input rate that is both load dependent and output rate dependent. The rightmost "tail" of the curve is well into the "thrashing area". The term "thrashing" while frequently applied erroneously to other types of problems, is generally used to refer specifically to a problem that may occur in memory management, namely the point at which processing efficiency seriously degrades during an attempt to overcommit main memory. This same problem can occur as well with buffering, and caching, so I will apply the term in this discussion as well.

Before proceeding, let's take the time to define the general concept of some of the terms we will be using here:

INPUT RATE: the number of requests made to the subsystem.

OUTPUT RATE: the number of requests satisfied by the subsystem.

SERVICE RATE: the rate at which requests are satisfied per unit of time.

LEVEL of CONCURRENCY: the number of processes which are being concurrently satisfied (multiprogramming level in memory case)

The horizontal line represents a limit that is imposed upon the system often to prevent the system from degrading any further. Memory management systems generally have such a limit built in, or tuneable, to

prevent "thrashing" (such as the MPE ":TUNE" command). Potentially, there are many other ways of limiting overcommitment (a job limit, a disc bottleneck, etc.).

Given such a characteristic curve here, there are at most three points at which the input rate is equal to the output rate (sometimes there would be only one or two). We have labeled these critical points "A", "B", and "C".

Points "A" and "C" are stable points on the curve (existing at locations on the curve which have positive slopes). They are stable from the standpoint that an increase in the input rate will be met by a larger increase in the service rate causing the input rate to tend to decline back to the stable point (equilibrium). Likewise a decrease in the input rate will be met by a larger decrease in the service rate which will tend to cause the input rate to increase back to the point of stability (equilibrium).

Point "B", however, is a point of instability (exists at a location on the curve which has a negative slope). An increase in the input rate at this point will be met by a decrease in the service rate which will cause a still further increase in the input rate. Likewise, a decrease in the input rate at this point will be met by an increase in the service rate which will tend to cause a further decrease in the input rate. Point "B" is a point at which the system will resist almost as if repelled, and the closer it approaches the stronger the repulsion.

What is very interesting about this point of instability (and the whole point of this discussion) is that in order for the system to cross it, the system needs a very large or sudden increase in the input rate, a "kick". Once it has been "kicked" past this point it will frequently be in a less productive region at a point of stability ("C" on our curve). After it has been "kicked" past the point of instability to a new point of stability, frequently even when processing demand starts diminishing, there will be no event that will "kick" it back past the point of instability, so the system will tend to stay in the often less productive area of point "C".

I have found situations where, with software disc caching, this seems to happen with some regularity (if we allow it). To avoid it, we programmatically monitor the system load and make adjustments to caching primarily to prevent entry into the "thrashing area", but more importantly (at least in terms of this discussion) provide the "kick" necessary to bring the system back to a stable area well outside of the "thrashing area".

CONCLUSION

We have covered quite a bit of ground here, while only skimming the surface. In our travels, we may have been able to dispel some of the common myths to relating to various subsystems, and replace some "old

rules of thumb" with "good common sense" based upon a better understanding of the underlying characteristics of the particular subsystem.

We have seen here throughout course of the paper, that the scheduling of the workload on the HP3000's ultimately becomes far more dependent upon the limiting resource(s) (the "bottleneck"). If we imagine a data base bottleneck, with batch jobs and interactive processing both accessing the data base, the batch job (because of zero "think time") will gain an unfair processing advantage over the interactive processes. The memory management algorithm will then tend to reinforce this unfair advantage by its "recent use" algorithm. The unfair advantage will be further reinforced by disc caching. The temporary reprioritization scheme utilized by IMAGE will prevent the I/O prioritizing scheme from supporting the high priority processes' I/O, further reinforcing the unfair advantage gained by the batch job. WHEN A BOTTLENECK EXISTS, EVERYTHING YOU EVER LEARNED ABOUT PRIORITY SCHEDULING MAY VERY WELL BECOME "INOPERABLE".

Bottlenecks have very important implications, not just from the slower response of those processes waiting for service at the bottleneck, but very often from the side effects that the bottleneck introduces.

Many countless hours of effort are spent trying to improve performance, but, if I may quote Alfredo Rego, "the path to expectation is paved with disappointment". Before we invest a single second in efforts to improve performance, we need to understand the system we are trying to improve, and we need to be able to determine specifically where to place our efforts in order to achieve the best "return on investment".

I have presented a very specific method for tracking the requirements for a very specific (and common) problem, that related to IMAGE data base processing. If we desire, we can limit our use of this information solely to that problem, but better yet, we'll recognize the need to develop specific methods for understanding various performance problems/consequences.

It is my hope that, with what has been presented here, others will be inspired to a higher level of sophistication in developing the methods necessary to best address HP3000 performance issues.

FOOTNOTES

¹Knuth [KNUTH d] provides a proof for a formula that arrives at a much better approximation, but as he points out, "[the formula] becomes useless when [the ratio of entries to the capacity] approaches one". To be safe we chose the less accurate approximation in [KNUTH c] which does not degrade as the file becomes full. In addition we might be cautioned that the standard deviation tends to be rather large over many occurrences of this particular problem, and it very definitely does not account for any special "clustering" problems. The formula is at best an approximation.

²"DBLOADNG" was written by Rick Berquist of American Management Systems and is contained in the INTEREX contributed library. "HOWMESSY" was written by David Greer of Robelle Consulting, Ltd., and is provided to their customers. Both programs report performance related structural information as found in an existing IMAGE data base. [GREEN b] explains the use of either program.

³I have that statement both orally from Peter J. Denning and (for those of you skeptics) in [DENNING].

⁴We are "bending the rules" somewhat on this one. To begin with we are treating processes working within this data base scenario as completely separate from the rest of the system. The most significant exception would be other I/O's to the disc drive(s) which the data base reside on; there certainly would be some queuing for other non-database I/O. But, if anything, that should make our results somewhat optimistic, under our assumptions.

Additionally, although it is usually appropriate to place "think time" into the calculation of response time (16 seconds), we did not do so here as the requirements for each transaction against the data base are so large that saturation is so quickly achieved and the system will start acting as if it were batch (no "think time"). The 300 milliseconds of cpu time was not considered either as it is so small in comparison to the overall transaction time that its effect becomes negligible, and it helps us to more successfully isolate it from other (non order entry) processing on the system. This would possibly make our estimates of response time a bit pessimistic, if ever so slightly, but only for few transactions per observation window.

The end results, I trust, will not be misleading in the insight that they make possible.

REFERENCES

- CARROLL: Carroll, Bryan, "MPE Disc Caching", INTEREX Proceedings of the 1985 Conference, INTEREX, Washington, D.C., 1985, pp 1-15.
- COURTOIS: Courtois, P.J., "Decomposability, Instabilities, and Saturation in Multiprogrammed Systems", COMMUNICATIONS of the ACM 18, 7 (July 1975), pp 371-377.
- DENNING: Denning, Peter, J., "Queuing Network Models", Encyclopedia of Computer Science and Engineering, 2nd ed., Van Nostrand Reinhold Company, New York, N.Y., 1983, p 1253.
- GREEN a: Green, Robert M.; Rego, F. Alfredo; White, Fred; Greer, David J.; Heidner, Dennis L., "The IMAGE/3000 HANDBOOK", Wordware, Seattle, Wash., 1984.
- GREEN b: Green, Robert M.; Rego, F. Alfredo; White, Fred; Greer, David J.; Heidner, Dennis L., "The IMAGE/3000 HANDBOOK", Wordware, Seattle, Wash., 1984, pp 257-265.
- HP: Hewlett-Packard Company, "Site Environmental Requirements for Disc/Tape drives", part no. 5955-3456, Hewlett-Packard Company, Boise, Idaho, 1984, p A-17.
- KNUTH a: Knuth, Donald E., "The Art of Computer Programming", Addison-Wesley Publishing Company, Reading, Mass., 1973, v 3, p 518.
- KNUTH b: Knuth, Donald E., "The Art of Computer Programming", Addison-Wesley Publishing Company, Reading, Mass., 1973, v 3, p 518.
- KNUTH c: Knuth, Donald E., "The Art of Computer Programming", Addison-Wesley Publishing Company, Reading, Mass., 1973, v 3, p 528.
- KNUTH d: Knuth, Donald E., "The Art of Computer Programming", Addison-Wesley Publishing Company, Reading, Mass., 1973, v 3, p 530.
- LAZOWSKA: Lazowska, Edward D.; Zajorjan, John; Graham, G. Scott; Sevcik, Kenneth C.; "Quantitative System Performance", Prentice-Hall, Inc., Englewood Cliffs, NJ, 1984.
- VOLOKH a: Volokh, Eugene, "Thoughts and Discourses on HP 3000 Software", Vesoft, Inc., Los Angeles, CA, 1984, p 162.
- VOLOKH b: Volokh, Eugene, "Thoughts and Discourses on HP 3000 Software", Vesoft, Inc., Los Angeles, CA, 1984, p 168.

Process Handling With Business BASIC
by
Jack Craig

BRIDGEWARE
501 McDonald Rd.
Aptos, CA. 95003

TABLE OF CONTENTS

- 1.0 INTRODUCTION
- 2.0 Business BASIC IMPROVES APPLICATION ACCESS TO PROCESS HANDLING
 - 2.1 THE SYSTEM COMMAND
 - 2.2 THE SYSTEMRUN COMMAND
- 3.0 INTERPROCESS COMMUNICATION
 - 3.1 INTERPROCESS COMMUNICATION OPTIONS
 - 3.11 MESSAGE FILES
 - 3.12 CIRCULAR FILES
 - 3.13 MPE FILES
 - 3.14 JOB CONTROL WORDS
 - 3.2 PROCESS IDENTIFICATION
- 4.0 PROCESS TREE MANIPULATION
 - 4.1 CREATING A PROCESS WITH THE INTERPRETER
 - 4.2 TERMINATING A PROCESS
- 5.0 APPLICATION SAMPLE
 - 5.1 SAMPLE DESCRIPTION
 - 5.11 SAMPLE LISTING - THE TESTSIZE UTILITY
 - 5.12 SAMPLE PROGRAM OUTPUT
- 6.0 CONCLUSION
- 7.0 REFERENCES

1.0 INTRODUCTION

The HP3000 is a multiprocessing computer. This means that it is able to handle many processes at once. The operating system manages processes at many levels. These levels range from the highest priority processes of the memory manager to the lowest level user process. The operating system has responsibility to keep track of what processes are beginning, what resources must be located and moved to memory, which processes are executing, and which are finishing. When a user runs a program, the system creates a process. This process is activated, begins to execute, and if there is not an infinite loop, finishes.

Process handling is managed by the system in a parent-child relationship. The primary or controlling process is the parent process. New processes are considered to be children of that parent process. This parent and one or more children processes are referred to as a process tree. There are two categories of process handling. The first is in-line process handling: a parent process begins a child, then goes to sleep until that child completes, then continues execution. The second is parallel process handling: a parent sets up one or more children processes, a child possibly sets up another child process, and they all are executing at the same time. This category of process handling uses some interprocess communication to control the state and function of the different processes.

In the past, process handling has only been used by experienced system programmers. This was due in great measure to the tedious nature of developing process handling programs. There are several aspects of process handling to be considered by the programmer. They are: creation, activation, termination, and interprocess communication. The creation and activation steps are accomplished via direct calls to intrinsics. The interprocess communication aspect is up to the creativity of the programmer and the specific application requirements.

It was the intrinsic calling aspect that posed a barrier for BASIC/3000 users. BASIC/3000 could not call most intrinsics. This required writing an SPL subroutine to call the intrinsic and a convoluted parameter passing operation to communicate with the calling BASIC program. This tended to limit the use of process handling to only the most demanding requirements. This paper presents HP Business BASIC and its capabilities to provide access to process handling as well as many options for interprocess communication.

2.0 Business BASIC IMPROVES APPLICATION ACCESS TO PROCESS HANDLING

Business BASIC provides several major features for application access to process handling. The first of these is provided by a built-in command to access the operating system. The second is provided by a built-in command to execute a child process. The parameters of this command provide for both in-line processes as well as parallel process handling.

2.1 THE SYSTEM COMMAND

The SYSTEM statement is a command to allow communication with the operating system. Its function is to send a string literal or quoted string to MPE. It also provides a ;STATUS=Mpe_err_num parameter to return the operating system's error or warning to the calling program. A sample of its use is:

```
10 SYSTEM "showjob";STATUS=Mpe_err
20 SYSTEM "file t;dev=tape"
30 My_command$="showme"
40 SYSTEM My_command$
```

2.2 THE SYSTEMRUN COMMAND

The SYSTEMRUN command allows the programmer to run a program as a child process. The program with the SYSTEMRUN is considered to be the parent process. Any subsequent SYSTEMRUN statements will create child processes related to the parent. The SYSTEMRUN command will accept either a string literal or a quoted string. It also has the ;STATUS=Mpe_err_num parameter that returns the job control word of the called process. This may then be tested for success or failure. The values returned in this parameter are defined in the system intrinsics manual. It has a NOSUSP parameter as well. When a SYSTEMRUN command is executed, the named program file is loaded and begins execution. If the NOSUSP parameter is not present the parent process is suspended until the child process is complete. If the NOSUSP parameter is specified, the child process is launched, but the execution of the parent process continues without further regard for the child processes' status. It is this scheme that allows for the parallel processing. This may require communication between the parent and its children or between the children.

3.0 INTERPROCESS COMMUNICATION

Interprocess communication is the most flexible and potentially the most vulnerable aspect of process handling. In-line process handling is the simplest to manage. There are no timing considerations for interprocess communication as the parent process launches the child and waits for the child to complete before the parent continues to execute. Parallel process handling requires a more elaborate communication scheme.

Messages sent that are not received, sent too late or too early, can cause a programmer to swear off process handling permanently. This can be especially frustrating in a system environment with fluxuating work loads. An interprocess communication scheme that works fine in a lightly loaded system could fail miserably in the heavy batch processing time frame. This requires the analyst to thoroughly consider all aspects of the intended communication method for the possibility of child process aborts, heavily loaded processors, and other environmental variables.

Interprocess communications may be sent in several directions: a parent process may send a message to a child process, a child process may send a message to its parent process. A child may also send messages to other children on the process tree. One process tree may communicate with another process tree. This requires the process identification number(PIN) of the target process. This intertree communication is subject to the MPE system security matrix.

3.1 INTERPROCESS COMMUNICATION OPTIONS

Methods of interprocess communication have changed over recent years. There are intrinsics provided to send and receive messages. SENDMAIL and RECEIVEMAIL have been available for some time. They, however, require the target's process identification number, and the programmer must provide a storage area for the messages sent. There are problems with this implementation in that the PIN was sometimes not easily available and the mailbox was not able to handle many messages at once. There are several methods now available that provide much greater flexibility, greater storage capacity, and in the case of message files, better performance.

3.11 MESSAGE FILES

The fastest method of interprocess communication is based on message files. Message files are faster primarily because they are memory resident. The message file stores records as the sender writes them into the file, and deletes them as the receiver reads them. Memory storage space is dynamically allocated as it is needed and released as messages are collected by the receiving process. This is good from a performance and resource usage point of view, but this method is very vulnerable to power failure and system halts. The Business BASIC function NUMREC(file_num) is useful to tell the calling program if there are records in the file to be read.

3.12 CIRCULAR FILES

Circular files provide an interesting twist in files usage. A circular file will never return an EOF condition. If you write enough records to fill the file, it will simply wrap around and begin writing records at the beginning of the file. This is a useful method if your application needs to be able to go back a few records once in a while. It is up to the programmer to track what record is being written on via a date/time

stamp or some other marking scheme. This is a method that will appeal to applications that want to recycle their disc space. This is also a good scheme if crash recovery is necessary.

3.13 MPE FILES

MPE files are sometimes useful for interprocess communication. The programmer may open the same file several times. This allows a process to write a record on a specific location with one process and concurrently read it in with another process. The caution here is that if the application adds records to the MPE file, and the file is filled, the process on attempting to add another record has no choice but to abort. With this possibility in mind the programmer may provide safeguards against this. Failure to do so may topple the entire process tree.

3.14 JOB CONTROL WORDS

Job control words(JCW) may sometimes provide an easy method of communicating numeric values from one process to another. JCW are common to a job or session. This means that if Process A sets My_JCW = 100, Process B may immediately use My_JCW with the certainty that its value is 100 as far as Process B is concerned. There are constraints with JCW usage. JCWs are implemented with a 16 bit word where the system uses the zeroeth or leftmost bit. This allows only numeric values that may be represented with 15 bits. Another consideration is that there is a limitation to the number of JCWs that may exist at once. The system intrinsics manual is a good reference for the correct procedure to setup a JCW, load it with some value, change that value, and deallocate that JCW.

3.2 PROCESS IDENTIFICATION

Interprocess communication via SENDMAIL and RECEIVEMAIL require a PIN. This may be obtained several ways. The intrinsic GETPROCID has been available for some time. Business BASIC provides an easier method with the built-in function called TASKID. This allows any process to be quickly identified and that PIN may be sent to any other process that requires it.

4.0 PROCESS TREE MANIPULATION

Process trees may be found in many application areas. Some of the most common are those used to circumvent the users' direct access to MPE. The parent process creates processes for users' sessions and subsequently controls all access via program control. This provides a very tight security scenario while tasking the programmer to recreate many of the functions of MPE. An excellent example of process tree manipulation may be found in the Business BASIC compiler. The compiler has a big job to do in that it must generate machine code to cope with all the eventualities that the resulting program will encounter at run-time,



without the luxury of knowing the run-time environment as the interpreter does. To spread out this task the compiler makes use of a background process. It communicates with this background process via message files. When the compile process is done, it checks that its child process has terminated and then gracefully terminates the parent process.

4.1 CREATING A PROCESS WITH THE INTERPRETER

Business BASIC has an interpreter as well as a compiler. The interpreter, being a program itself, can be called as a child process just as any other program. Several samples of this may be found in the Business BASIC sample program. The interpreter may call the interpreter, or another object code program. The interpreter uses several external files that may be redirected by the user. They are the BASCOM file, the BASIN file, and the BASOUT file. The BASCOM file is used to input commands or program lines to the interpreter. The BASIN file is used to accept data from input statements such as LINPUT, INPUT, etc. This file is handled differently if a JOINFORM or VPLUS form is active. The BASOUT file is used for program output such as PRINT. These files may be redirected with MPE file equations and passing a ;parm=x at run-time or systemrun-time. The parameter values required are:

| file to be redirected ----- | parameter value ----- |
|--------------------------------|--------------------------|
| BASCOM | 1 |
| BASIN | 2 |
| BASCOM + BASIN | 3 |
| BASOUT | 4 |
| BASCOM + BASOUT | 5 |
| BASIN + BASOUT | 6 |
| BASCOM + BASIN + BASOUT | 7 |

Another built-in convenience is the INFO\$ function. Its use is to return the contents of the run-time parameter ;info="something". The following is an example of the INFO\$ function:

```
10 DIM Catch_info$(80)
20 Catch_info$=INFO$
30 PRINT Catch_info$
```

When compiled and run as:

```
:run demo;info="* this is the info string *"
* this is the info string *
end of program
```

Note that this is relative to object code programs only as ;info= is a

run-time parameter. This remains useful to the interpreter. For example:

```
:file basout=*lp
:run hpbb.pub.sys;parm=4;info="run myprog"
```

In this sample, the output of myprog is redirected to the line printer while ;info= is passed to the BASCOM file and the interpreter runs myprog. An interesting option is found in the sample program when the utility creates program remark lines, adds them to a file, merges them with the sample to be tested, and saves the original program plus new lines to a file for compiling. I have used this technique to create and save a new program file as well as modify an existent file.

4.2 TERMINATING A PROCESS

The most desirable method to terminate a process is to have it execute an END or STOP command. This provides the programmer with a good warm feeling that all went well in the child process and ended as it should. Many programmers use the QUIT(Quit_var) command to terminate the process in the event of an unexpected error. The Quit_var will appear in the process abort message. This allows the programmer to put a QUIT(Quit_var) in several places in the program with different Quit_var values in each place. This provides a quick reference to the location of the error.

The system intrinsic KILL(PIN) is provided to flush processes existing on a process tree. The parent process may not be terminated until all the children are terminated. The programmer may design the parent process to keep a list of the PINs for each new child so that in the event that the parent has a hard abort, kill commands for each child may be generated prior to ending the process.

5.0 APPLICATION SAMPLE

The following sample was selected because of the variety of process handling options used. This sample makes use of in-line process handling. The program and resulting sample output are just that - samples. The program logic is valid and compilable. The sample output represents valid relationships with respect to object code generated, but does not reflect the actual amount of code generated with a production compiler.

5.1 SAMPLE DESCRIPTION

The sample program is simple to operate. It prompts the user for a source code program filename. This may be either a BSAVE file or an ASCII file. The user is then prompted for display of analysis by line number, by segment, and would they like a copy on the system printer. At this point, the program takes over and does its thing. The first step is to call the interpreter to convert the user program to an ASCII format. The INDENT command is used to align the program source code. This ASCII

file is then read to determine where the MAIN stops and the subunits begin. In Business BASIC a program is made up of one MAIN unit and possibly some subunits. Multi-line defined functions are treated as subunits for the purpose of this analysis. The TESTSIZE program locates the beginning of each subunit and creates a new remark line to be added just before each subunit and at the end of the program. The next step is to merge these new program lines and some compiler options and save the resulting new program in a file. The compiler is then called to compile this program. If there are compiler errors, the program reports this and cleans up files created. If it compiles, the segmenter is called to review the resulting user subprogram library(USL) file. The program reads the resulting USL file for segments and procedures and their respective code sizes. The USL file lists code sizes in octal, so a decimal conversion is performed. The program has now performed five different process handling steps with an in-line method. The results are now printed out based on the user directives and the program cleans up its files.

5.11 SAMPLE LISTING - THE TESTSIZE UTILITY

This listing is a sample of Business BASIC using process handling.

```

100 ! TESTSIZE - a Business BASIC/3000 development tool
110 ! utility to measure object code sizes - by jack craig 3/13/86
120 ! last updated - 5/12/86
130 !
140 COPTION SEGMENT="mainseg"
150 OPTION BASE 1
160 DIM Command$(80),Input_buf$(80),Work_buf$(5)[14],Dashed_line$(40)
170 DIM Line_detail$(1),Segment_detail$(1),Line_printer$(1)
180 DIM Proc_list$(150)[22],Segment_name$(8),Full_file_name$(36)
190 INTEGER Data_array(600,4),Da_index,Off_set,Cum_off_set,Total_cod
200 INTEGER Usl_code_proc(150),Prog_code_proc(150),For_display(150)
210 INTEGER Outer_block_code
220 Dashed_line$=RPT$("-",40)
230 ! input filename to be tested
240 LINPUT "source file to measure: ";Full_file_name$(1)
250 IF LEN(TRIM$(Full_file_name$))=0 THEN GOTO 300
260 ! does it exist ?? use interpreter to convert it
270 ASSIGN #1 TO Full_file_name$,STATUS=Open_err
280 IF Open_err THEN
290 PRINT " unable to open filename: ";Full_file_name$;" error# "
Open_err
300 STOP

```

```

310 ENDIF
320 ASSIGN * TO #1
330 ! offer user options
340 LINPUT "analysis by line number ? ";Line_detail$[1]
350 IF POS("Yy",Line_detail$) THEN Line_detail$="y" ELSE Line_detail
360 LINPUT "analysis by segment ? ";Segment_detail$[1]
370 IF POS("Yy",Segment_detail$) THEN Segment_detail$="y" ELSE &
Segment_detail$="n"
380 LINPUT "copy options to printer ? ";Line_printer$[1]
390 IF POS("Yy",Line_printer$) THEN Line_printer$="y" ELSE Line_prin
n"
400 IF Line_printer$="y" THEN
410 SYSTEM "file testlist;dev=pp;env=elite.env2680a.sys"
420 ENDIF
430 CALL Cleanup ! cleanup from previous failed runs
440 CREATE ASCII "script"
450 ! need to instrument user program so i can see offsets to end/su
460 ASSIGN #1 TO "script",STATUS=Open_err
470 PRINT #1;"get "+Full_file_name$
480 PRINT #1;"indent"
490 PRINT #1;"save list putprobe"
500 PRINT #1;"exit"
510 ASSIGN * TO #1
520 SYSTEM "file bascom=script"
530 Command$="hpbb.pub.sys;lib=g;parm=1;stdlist=$null"
540 Home_clear: PRINT '27"h"27"J"
550 PRINT "inserting test probes in source code"
560 PRINT "dont worry, i'll use my own copy."
570 SYSTEMRUN Command$;STATUS=Mpe_err
580 ! open putprobfile and read it
590 ASSIGN #1 TO "putprobe",STATUS=Open_err
600 ON END #1 GOTO 750
610 Period=POS(Full_file_name$,".")
620 IF Period THEN Full_file_name$=Full_file_name$[1,Period-1]
630 Da_index=1;Record_num=1;Num_subs=0
640 MAT Data array=ZER
650 LINPUT #1,Record_num;Input_buf$[1]
660 ! test input
670 CALL Decide_to_probe(Input_buf$,Yes_or_no)
680 IF Yes_or_no THEN
690 ! break out line# and save line # plus sub level
700 Data array(Da_index,1)=VAL(Input_buf$[1,7])
710 Da_index=Da_index+1;Num_subs=Num_subs+1
720 ENDIF
730 Record_num=Record_num+1
740 GOTO 650
750 ! now you know whats where, probe it
760 Da_index=Da_index-1
770 CALL Reset_script
780 ASSIGN #1 TO "script",STATUS=Open_err
790 PRINT #1;"get putprobe"

```

```

800 ! if some subs, probe ahead of sub statement
810 IF Num_subs>0 THEN
820     FOR I=1 TO Da_index
830         PRINT #1;VAL$(Data_array(I,1)-1)+" ! test probe ****"
840     NEXT I
850 ENDIF
860 PRINT #1;"999999 ! test probe ****"
870 PRINT #1;"resave"
880 PRINT #1;"exit"
890 ASSIGN * TO #1
900 ! now do it
910 SYSTEMRUN Command$;STATUS=Mpe_err
920 ! probe insertion complete/ reset
930 MAT Data_array=ZER
940 Da_index=1;Num_subs=0
950 CALL Reset_script
960 ASSIGN #1 TO "script",STATUS=Open_err
970 PRINT #1;"get putprobe"
980 PRINT #1;"1 global coption label tables,uslinit"
990 PRINT #1;"save sourcef"
1000 PRINT #1;"save list copyf"
1010 PRINT #1;"exit "
1020 ASSIGN * TO #1
1030 Home_clear: PRINT '27"h"'27"j"
1040 PRINT "converting format to BSAVE"
1050 SYSTEMRUN Command$;STATUS=Mpe_err
1060 SYSTEM "reset bascom"
1070 ! set up files for compiler, input output
1080 SYSTEM "file bbcin=sourcef"
1090 ! set up a usl file for some good numbers
1100 SYSTEM "build myusl;code=usl;disc=2000,1"
1110 CREATE ASCII "savecomp",FILESIZE=2000
1120 CREATE ASCII "saveusl",FILESIZE=2000
1130 SYSTEM "file bbclist=savecomp"
1140 SYSTEM "file bbcusl=myusl"
1150 ! call compiler
1160 Command$="hpbbcmp.pub.sys;lib=g;parm=-1;stdlist=$null"
1170 Home_clear: PRINT '27"h"'27"j"
1180 PRINT "compiling source file: ";Full_file_name$
1190 SYSTEMRUN Command$;STATUS=Mpe_err
1200 ! test for compiler problems
1210 IF Mpe_err THEN
1220     CALL Handle_compiler_error(Mpe_err,Command$)
1230     PRINT Command$
1240     WAIT (2)
1250     IF Mpe_err>32767 THEN Cleanup
1260 ENDIF
1270 ! read back the code offsets/ trace by line#
1280 ASSIGN #1 TO "savecomp",STATUS=Open_err
1290 ON END #1 GOTO 1500
1300 LINPUT #1;Input_buf$[1]

```

```

1310 CALL Decide_to_probe(Input_buf$,Yes_or_no)
1320 IF Yes_or_no THEN Num_subs=Num_subs+1
1330 IF Input_buf$[7;1]<>"=" THEN 1300
1340 ! extract offsets data_array
1350 Index=1
1360 FOR I=1 TO 65 STEP 16
1370   Work_buf$(Index)=Input_buf$[I;14];Index=Index+1
1380 NEXT I
1390 FOR I=1 TO 5
1400   IF LEN(TRIM$(Work_buf$(I)))=0 THEN 1480
1410   Data_array(Da_index,1)=Num_subs
1420   Off_set=VAL(Work_buf$(I)[8;14])
1430   Data_array(Da_index,2)=Off_set
1440 ! change base from octal to decimal
1450   CALL Octal_to_decimal(Off_set)
1460   Data_array(Da_index,3)=Off_set
1470   Da_index=Da_index+1
1480 NEXT I
1490 GOTO 1300
1500 ASSIGN * TO #1
1510 ! read usl file for content
1520 CALL Reset_script
1530 ASSIGN #1 TO "script"
1540 PRINT #1;"usl myusl"
1550 PRINT #1;"listusl"
1560 PRINT #1;"exit"
1570 ASSIGN * TO #1
1580 ! process the segmenter
1590 Home_clear: PRINT '27"h" '27"j"
1600 PRINT "reviewing results with segmenter"
1610 Command$="segdvr.pub.sys;stdin=script;stdlist=saveusl"
1620 SYSTEMRUN Command$;STATUS=Mpe_err
1630 ASSIGN #1 TO "saveusl",STATUS=Open_err
1640 ON END #1 GOTO 1900
1650 POSITION #1;7
1660 Num_proc=1
1670 LINPUT #1;Input_buf$[1]
1680 ! header/segmentname/procedure ??
1690 IF LEN(TRIM$(Input_buf$))=0 THEN 1670
1700 IF Input_buf$[1,9]="FILE SIZE" THEN 1900
1710 IF Input_buf$[4,6]="OB" THEN
1720   Outer_block_code=VAL(Input_buf$[20;5])
1730   CALL Octal_to_decimal(Outer_block_code)
1740   GOTO 1670
1750 ENDIF
1760 IF Input_buf$[1,3]=" " THEN
1770   Prime_prime=POS(Input_buf$[1,14],",")
1780   IF Prime_prime AND POS(Input_buf$," FN") THEN
1790     Input_buf$[4,14]="**"+Input_buf$[4,Prime_prime-1]
1800   ENDIF
1810   Proc_list$(Num_proc)[1,14]=Input_buf$[4;14]

```

```

1820 Proc_list$(Num_proc)[15,22]=Segment_name$
1830 Usl_code_proc(Num_proc)=VAL(Input_buf$[20;5])
1840 Num_proc=Num_proc+1
1850 GOTO 1670
1860 ENDIF
1870 Segment_name$=Input_buf$[1,8]
1880 IF Segment_name$[1,4]="SEG" THEN Segment_name$="MAINSEG"
1890 GOTO 1670
1900 ASSIGN * TO #1
1910 Num_proc=Num_proc-1;Cum_off_set=Outer_block_code
1920 ! lets cleanup for the segmenter
1930 Proc_list$(Num_proc)[1,14]=UPC$(Full_file_name$)
1940 ! convert segmenter code to decimal
1950 FOR J=1 TO Num_proc
1960 CALL Octal_to_decimal(Usl_code_proc(J))
1970 Cum_off_set=Cum_off_set+Usl_code_proc(J)
1980 NEXT J
1990 ! calculate code space by line
2000 Da_index=Da_index-1
2010 FOR J=1 TO Da_index-1
2020 IF Data_array(J,1)<>Data_array(J+1,1) THEN
2030 Data_array(J,4)=0
2040 GOTO 2070
2050 ENDIF
2060 Data_array(J,4)=Data_array(J+1,3)-Data_array(J,3)
2070 NEXT J
2080 ! now lines/code per subunit
2090 FOR J=1 TO Da_index
2100 Prog_code_proc(Data_array(J,1)+1)=Prog_code_proc(Data_array(J
1)+1)+Data_array(J,4)
2110 NEXT J
2120 ! now move for_display() for single line def functions
2130 Index_prog=1;Index_usl=Num_proc
2140 IF Index_prog>Num_subs+1 THEN 2230
2150 ! is this a single line function ??
2160 IF Proc_list$(Index_usl)[1,2]="*" THEN
2170 Index_usl=Index_usl-1
2180 GOTO 2160
2190 ENDIF
2200 For_display(Index_usl)=Prog_code_proc(Index_prog)
2210 Index_usl=Index_usl-1;Index_prog=Index_prog+1
2220 GOTO 2140
2230 ! on to display
2240 Home_clear: PRINT '27"h"27"J"
2250 IF Line_printer$="y" THEN COPY ALL OUTPUT TO "testlist"
2260 ! list output by line with detail
2270 IF Line_detail$="n" THEN 2530
2280 ASSIGN #1 TO "copyf",STATUS=Open_err
2290 IF Open_err THEN
2300 PRINT "unable to open copyf, error: ";Open_err
2310 GOTO 2810

```

```

2320 ENDIF
2330 ON END #1 GOTO 2520
2340 IMAGE 40A/40A/40A
2350 PRINT USING 2340;Dashed_line$,"code size analysis by line number
Dashed_line$
2360 IMAGE "words"2X"offset"2X"program logic"
2370 PRINT USING 2360
2380 Record_num=1;Index=1
2390 LINPUT #1,Record_num;Input_buf$[1]
2400 IF Input_buf$[11,20]="test probe" OR VAL(Input_buf$[1,7])=1 THEN
2410 IMAGE 4D,X"%5D,68A
2420 PRINT USING 2410;Data_array(Index,4),Data_array(Index,2),Input_b
68]
2430 Record_num=Record_num+1
2440 IF Input_buf$[LEN(RTRIM$(Input_buf$));1]="&" THEN
2450 LINPUT #1,Record_num;Input_buf$[1]
2460 IMAGE 11X,68A
2470 PRINT USING 2460;Input_buf$[1,68]
2480 GOTO 2430
2490 ENDIF
2500 Index=Index+1
2510 GOTO 2390
2520 ASSIGN * TO #1
2530 ! list procedures by size
2540 IF Segment_detail$="n" THEN 2810
2550 PRINT USING 2340;Dashed_line$,"code size analysis by procedure/&
subunit",Dashed_line$
2560 IMAGE "segment name"17X"total code"7X"logic code"4X"overhead cod
procedure name"12X"generated"8X"generated"5X"generated"
2570 PRINT USING 2560
2580 IMAGE 3X"OUTER BLOCK",15X5D,2X"words"/
2590 PRINT USING 2580;Outer_block_code
2600 PRINT Proc_list$(Num_proc)[15,22] ! output the first segment
2610 Total_code=0
2620 FOR J=Num_proc TO 2 STEP -1
2630 IMAGE 3X14A,12X5D,2X"words",X9D2X"words",X9D2X"words"
2640 Dif=Us1_code_proc(J)-For_display(J)
2650 PRINT USING 2630;Proc_list$(J),Us1_code_proc(J),For_display(J)
2660 Total_code=Total_code+Us1_code_proc(J)
2670 IMAGE 29X5D,2X"words/segment"/8A
2680 IF Proc_list$(J)[15,22]<>Proc_list$(J-1)[15,22] THEN
2690 PRINT USING 2670;Total_code,Proc_list$(J-1)[15,22]
2700 Total_code=0
2710 ENDIF
2720 NEXT J
2730 Total_code=Total_code+Us1_code_proc(1)
2740 Dif=Us1_code_proc(1)-For_display(1)
2750 IMAGE 3X14A,12X5D,2X"words",X9D2X"words",X9D2X"words"
2760 PRINT USING 2630;Proc_list$(1),Us1_code_proc(1),For_display(1),D
2770 IMAGE 29X5D,2X"words/segment"
2780 PRINT USING 2770;Total_code

```

```

2790 IMAGE /4X"Total code generated: ",8D,2X"words plus stt code"
2800 PRINT USING 2790;Cum_off_set
2810 Cleanup: CALL Cleanup
2820 IF Line_printer$="y" THEN COPY ALL OUTPUT TO DISPLAY
2830 END
2840 SUB Octal_to_decimal(INTEGER Parameter)
2850   COPTION SEGMENT="subseg"
2860   INTEGER Final_decimal
2870   Final_decimal=0;Local_string$=VAL$(Parameter)
2880   Len_of_string=LEN(TRIM$(Local_string$))
2890   FOR I=1 TO Len_of_string
2900     Final_decimal=Final_decimal*8+VAL(Local_string${I;1})-VAL(
2910     NEXT I
2920   Parameter=Final_decimal
2930 SUBEND
2940 SUB Reset_script
2950   SYSTEM "file scr=script;save"
2960   SYSTEM "purge *scr"
2970 SUBEND
2980 SUB Cleanup
2990   PURGE "sourcef";STATUS=Purge_err
3000   PURGE "savecomp";STATUS=Purge_err
3010   PURGE "saveusl";STATUS=Purge_err
3020   PURGE "copyf";STATUS=Purge_err
3030   PURGE "myusl";STATUS=Purge_err
3040   PURGE "script";STATUS=Purge_err
3050   PURGE "putprobe";STATUS=Purge_err
3060 SUBEND
3070 SUB Handle_compiler_error(Err_from_mpe,Return_comment$)
3080   IF Err_from_mpe>32767 THEN
3090     Return_comment$="fatal error during compile occurred..."
3100     GOTO 3130
3110   ENDIF
3120   Return_comment$="compiler warnings occurred during compile..."
3130 SUBEND
3140 SUB Decide_to_probe(Input_buf$,Yes_or_no)
3150 ! is it a sub ??
3160   Array_index=POS(Input_buf$," SUB ")
3170   IF Array_index=0 THEN 3230
3180   IF POS(Input_buf${Array_index-3;10},"GET SUB") OR POS(Input_b
Array_index-3;10],"DEL SUB") THEN 3230
3190   Num_value=NUM(Input_buf${Array_index+5;1])
3200   IF Num_value<65 OR Num_value>90 THEN 3270
3210   IF LEN(TRIM$(Input_buf${9,Array_index})) AND NOT POS(Input_bu
Array_index],":") THEN 3270
3220   GOTO 3290
3230 ! is it a defined multiline function ??
3240   Array_index=POS(Input_buf$,"FN")+2
3250   Num_value=NUM(Input_buf${Array_index+2;1])
3260   IF POS(Input_buf$," DEF ") AND POS(Input_buf$," FN") AND NOT
Input_buf$,"=") AND (Num_value>64 AND Num_value<91) THEN 3290

```



```

3270   Yes_or_no=0
3280   GOTO 3300
3290   Yes_or_no=1
3300 SUBEND

```

5.12 SAMPLE PROGRAM OUTPUT

This output is a sample generated by running the preceding utility on itself. The first part is a display of code generated by line number. This is based on the octal code offsets generated by the compiler. Note that remarks do not generate any code. Note that subunit declarations do not generate any code. They do generate addresses to which program control may be transferred. The first column is the amount of code generated by line in words decimal. The second column is the octal offsets generated by the compiler. The last column is part of the actual program logic. This is truncated to fit on the page. The last part of the output is a summary by segment and procedure. The segments in the program are listed with the procedures present within those segments. The program logic generates a certain amount of code. The difference between that and the resulting code total found in the USL is called the overhead. The overhead is a small amount of code space used at run time.

```

-----
code size analysis by line number
-----

```

```

words  offset  program logic
  0    %375   100 ! TESTSIZE - a Business BASIC/3000 development tool
  0    %375   110 ! utility to measure code segments - by jack craig
  0    %375   120 !
  0    %375   130 !
  0    %375   140 COPTION SEGMENT="mainseg"
  0    %375   150 COPTION NO RANGE CHECKING,NO SET ERRL,NO REDIM,NO ERR
  0    %375   160 OPTION BASE 1
  0    %375   170 DIM Command${80},Input_buf${80},Work_buf$(5)[14],Dash
  0    %375   180 DIM Line_detail$[1],Segment_detail$[1],Line_printer$[
  0    %375   190 DIM Proc_list$(150)[22],Segment_name$(8),Full_file_na
  0    %375   200 INTEGER Data_array(600,4),Da_index,Off_set,Cum_off_se
  0    %375   210 INTEGER Us1_code_proc(150),Prog_code_proc(150),For_di
  0    %375   220 INTEGER Outer_block_code
 56   %375   230 Dashed_line$=RPT$("-",40)
  0    %465   240 ! input filename to be tested
 20   %465   250 LINPUT "source file to measure: ";Full_file_name$[1
 50   %511   260 IF LEN(TRIM$(Full_file_name$))=0 THEN GOTO 310
  0    %573   270 ! does it exist ?? use interpreter to convert it
 18   %573   280 ASSIGN #1 TO Full_file_name$,STATUS=Open_err
  6    %615   290 IF Open_err THEN
 70   %623   300 PRINT " unable to open filename: ";Full_file_name$
        Open_err
  2    %731   310 STOP
  0    %733   320 ENDIF
 18   %733   330 ASSIGN * TO #1

```

```

0 %755 340 ! offer user options
20 %755 350 LINPUT "analysis by line number ? ";Line_detail$[1]
64 %1001 360 IF POS("Yy",Line_detail$) THEN Line_detail$="y" ELSE
20 %1101 370 LINPUT "analysis by segment ? ";Segment_detail$[1]
68 %1125 380 IF POS("Yy",Segment_detail$) THEN Segment_detail$="y"
Segment_detail$="n"
20 %1231 390 LINPUT "copy options to printer ? ";Line_printer$[1]
68 %1255 400 IF POS("Yy",Line_printer$) THEN Line_printer$="y" ELS
n"
47 %1361 410 IF Line_printer$="y" THEN
18 %1440 420 SYSTEM "file testlist;dev=pp;env=elite.env2680a.sy
0 %1462 430 ENDIF
8 %1462 440 CALL Cleanup ! cleanup from previous failed runs
33 %1472 450 CREATE ASCII "script"
0 %1533 460 ! need to instrument user program so i can see offset
27 %1533 470 ASSIGN #1 TO "script",STATUS=Open_err
71 %1566 480 PRINT #1;"get "+Full_file_name$
32 %1675 490 PRINT #1;"indent"
33 %1735 500 PRINT #1;"save list putprobe"
32 %1776 510 PRINT #1;"exit"
14 %2036 520 ASSIGN * TO #1
16 %2054 530 SYSTEM "file bascom=script"
20 %2074 540 Command$="hpbbs.pub.sys;lib=g;parm=1;stdlist=$null"
27 %2120 550 Home_clear=PRINT '27"n"'27"j"
21 %2153 560 PRINT "inserting test probes in source code"
23 %2200 570 PRINT "dont worry, i'll use my own copy."
22 %2227 580 SYSTEMRUN Command$,STATUS=Mpe_err
0 %2255 590 ! open putprobfile and read it
24 %2255 600 ASSIGN #1 TO "putprobe",STATUS=Open_err
4 %2305 610 ON END #1 GOTO 760
23 %2311 620 Period=POS(Full_file_name$,".")
114 %2340 630 IF Period THEN Full_file_name$=Full_file_name$[1,Peri
10 %2522 640 Da_index=1;Record_num=1;Num_subs=0
19 %2534 650 MAT Data_array=ZER
49 %2557 660 LINPUT #1,Record_num;Input_buf$[1]
0 %2640 670 ! test input
10 %2640 680 CALL Decide_to_probe(Input_buf$,Yes_or_no)
6 %2652 690 IF Yes_or_no THEN
0 %2660 700 ! break out line# and save line # plus sub level
113 %2660 710 Data_array(Da_index,1)=VAL(Input_buf$[1,7])
15 %3041 720 Da_index=Da_index+1;Num_subs=Num_subs+1
0 %3060 730 ENDIF
11 %3060 740 Record_num=Record_num+1
1 %3073 750 GOTO 660
0 %3074 760 ! now you know whats where, probe it
4 %3074 770 Da_index=Da_index-1
8 %3100 780 CALL Reset_script
24 %3110 790 ASSIGN #1 TO "script",STATUS=Open_err
45 %3140 800 PRINT #1;"get putprobe"
0 %3215 810 ! if some subs, probe ahead of sub statement
6 %3215 820 IF Num_subs>0 THEN

```

```

19 %3223      830      FOR I=1 TO Da_index
119 %3246      840          PRINT #1,VAL$(Data_array(I,1)-1)+" ! test probe
43  %3435      850      NEXT I
0   %3510      860      ENDIF
35  %3510      870      PRINT #1;"999999 ! test probe ***"
32  %3553      880      PRINT #1;"resave"
28  %3613      890      PRINT #1;"exit"
14  %3647      900      ASSIGN * TO #1
0   %3665      910      ! now do it
20  %3665      920      SYSTEMRUN Command$,STATUS=Mpe_err
0   %3711      930      ! probe insertion complete/ reset
19  %3711      940      MAT Data_array=ZER
6   %3734      950      Da_index=1;Num subs=0
8   %3742      960      CALL Reset script
24  %3752      970      ASSIGN #1 TO "script",STATUS=Open_err
45  %4002      980      PRINT #1;"get putprobe"
37  %4057      990      PRINT #1;"1 global coption label tables,uslinit"
42  %4124      1000     PRINT #1;"save sourcef"
36  %4176      1010     PRINT #1;"save list copyf"
33  %4242      1020     PRINT #1;"exit "
14  %4303      1030     ASSIGN * TO #1
23  %4321      1040     Home_clear: PRINT '27"h"'27"J"
21  %4350      1050     PRINT "converting format to BSAVE"
17  %4375      1060     SYSTEMRUN Command$,STATUS=Mpe_err
21  %4416      1070     SYSTEM "reset bascom"
0   %4443      1080     ! set up files for compiler, input output
16  %4443      1090     SYSTEM "file bbcin=sourcef"
0   %4463      1100     ! set up a usl file for some good numbers
16  %4463      1110     SYSTEM "build myusl;code=usl;disc=2000,1"
31  %4503      1120     CREATE ASCII "savecomp",FILESIZE=2000
35  %4542      1130     CREATE ASCII "saveusl",FILESIZE=2000
18  %4605      1140     SYSTEM "file bbclist=savecomp"
24  %4627      1150     SYSTEM "file bbcusl=myusl"
0   %4657      1160     ! call compiler
20  %4657      1170     Command$="hpbbcmp.pub.sys;lib=g;parm=-1;stdlist=$null
20  %4703      1180     Home_clear: PRINT '27"h"'27"J"
44  %4727      1190     PRINT "compiling source file: ";Full_file_name$
17  %5003      1200     SYSTEMRUN Command$,STATUS=Mpe_err
0   %5024      1210     ! test for compiler problems
6   %5024      1220     IF Mpe_err THEN
10  %5032      1230         CALL Handle_compiler_error(Mpe_err,Command$)
14  %5044      1240         PRINT Command$
11  %5062      1250         WAIT (2)
7   %5075      1260         IF Mpe_err>32767 THEN Cleanup
0   %5104      1270     ENDIF
0   %5104      1280     ! read back the code offsets/ trace by line#
24  %5104      1290     ASSIGN #1 TO "savecomp",STATUS=Open_err
4   %5134      1300     ON END #1 GOTO 1510
27  %5140      1310     LINPUT #1;Input_buf$[1]
10  %5173      1320     CALL Decide_to_probe(Input_buf$,Yes_or_no)
20  %5205      1330     IF Yes_or_no THEN Num_subs=Num_subs+1

```

```

118 %5231 1340 IF Input_buf$[7;1]<>"=" THEN 1310
0 %5417 1350 ! extract offsets data_array
4 %5417 1360 Index=1
18 %5423 1370 FOR I=1 TO 65 STEP 16
149 %5445 1380 Work_buf$(Index)=Input_buf$[I;14];Index=Index+1
29 %5672 1390 NEXT I
23 %5727 1400 FOR I=1 TO 5
47 %5756 1410 IF LEN(TRIM$(Work_buf$(I)))=0 THEN 1490
33 %6035 1420 Data_array(Da_index,1)=Num_subs
113 %6076 1430 Off_set=VAL(Work_buf$(I)[8,14])
13 %6257 1440 Data_array(Da_index,2)=Off_set
0 %6274 1450 ! change base from octal to decimal
9 %6274 1460 CALL Octal_to_decimal(Off_set)
13 %6305 1470 Data_array(Da_index,3)=Off_set
4 %6322 1480 Da_index=Da_index+1
29 %6326 1490 NEXT I
15 %6363 1500 GOTO 1310
14 %6402 1510 ASSIGN * TO #1
0 %6420 1520 ! read usl file for content
8 %6420 1530 CALL Reset_script
26 %6430 1540 ASSIGN #1 TO "script"
30 %6462 1550 PRINT #1;"usl myusl"
34 %6520 1560 PRINT #1;"listusl"
32 %6562 1570 PRINT #1;"exit"
14 %6622 1580 ASSIGN * TO #1
0 %6640 1590 ! process the segmenter
20 %6640 1600 Home_clear: PRINT '27"h" '27"J"
21 %6664 1610 PRINT "reviewing results with segmenter"
26 %6711 1620 Command$="segdvr.pub.sys;stdin=script;stdlist=saveusl
17 %6743 1630 SYSTEMRUN Command$,STATUS=Mpe_err
30 %6764 1640 ASSIGN #1 TO "saveusl",STATUS=Open_err
4 %7022 1650 ON END #1 GOTO 1910
9 %7026 1660 POSITION #1;7
4 %7037 1670 Num_proc=1
19 %7043 1680 LINPUT #1;Input_buf$[1]
0 %7066 1690 ! header/segmentname/procedure ??
37 %7066 1700 IF LEN(TRIM$(Input_buf$))=0 THEN 1680
127 %7133 1710 IF Input_buf$[1,9]="FILE SIZE" THEN 1910
125 %7332 1720 IF Input_buf$[4,6]="OB" THEN
95 %7527 1730 Outer_block_code=VAL(Input_buf$[20;5])
9 %7666 1740 CALL Octal_to_decimal(Outer_block_code)
1 %7677 1750 GOTO 1680
0 %7700 1760 ENDIF
119 %7700 1770 IF Input_buf$[1,3]=" " THEN
99 %10067 1780 Prime_prime=POS(Input_buf$[1,14],"'")
29 %10232 1790 IF Prime_prime AND POS(Input_buf$," FN") THEN
155 %10267 1800 Input_buf$[4,14]="*" + Input_buf$[4,Prime_prime-
0 %10522 1810 ENDIF
113 %10522 1820 Proc_list$(Num_proc)[1,14]=Input_buf$[4;14]
54 %10703 1830 Proc_list$(Num_proc)[15,22]=Segment_name$
112 %10771 1840 Us1_code_proc(Num_proc)=VAL(Input_buf$[20;5])

```

```

11 %11151 1850 Num_proc=Num_proc+1
1 %11164 1860 GOTO 1680
0 %11165 1870 ENDIF
83 %11165 1880 Segment_name$=Input buf$[1,8]
154 %11310 1890 IF Segment_name$[1,4]="SEG" THEN Segment_name$="MAIN
1 %11542 1900 GOTO 1680
21 %11543 1910 ASSIGN * TO #1
13 %11570 1920 Num_proc=Num_proc-1;Cum_off_set=Outer_block_code
0 %11605 1930 ! lets cleanup for the segmenter
69 %11605 1940 Proc_list$(Num_proc)[1,14]=UPC$(Full_file_name$)
0 %11712 1950 ! convert segmenter code to decimal
18 %11712 1960 FOR J=1 TO Num_proc
24 %11734 1970 CALL Octal_to_decimal(Us1_code_proc(J))
42 %11764 1980 Cum_off_set=Cum_off_set+Us1_code_proc(J)
29 %12036 1990 NEXT J
0 %12073 2000 ! calculate code space by line
7 %12073 2010 Da_index=Da_index-1
21 %12102 2020 FOR J=1 TO Da_index-1
62 %12127 2030 IF Data_array(J,1)<>Data_array(J+1,1) THEN
24 %12225 2040 Data_array(J,4)=0
8 %12255 2050 GOTO 2080
0 %12265 2060 ENDIF
75 %12265 2070 Data_array(J,4)=Data_array(J+1,3)-Data_array(J,3)
29 %12400 2080 NEXT J
0 %12435 2090 ! now lines/code per subunit
27 %12435 2100 FOR J=1 TO Da_index
84 %12470 2110 Prog_code_proc(Data_array(J,1)+1)=Prog_code_proc(D
1)+1)+Data_array(J,4)
29 %12614 2120 NEXT J
0 %12651 2130 ! now move for_display() for single line def function
11 %12651 2140 Index_prog=1;Index_us1=Num_proc
11 %12664 2150 IF Index_prog>Num_subs+1 THEN 2240
0 %12677 2160 ! is this a single line function ??
148 %12677 2170 IF Proc_list$(Index_us1)[1,2]="**" THEN
11 %13123 2180 Index_us1=Index_us1-1
1 %13136 2190 GOTO 2170
0 %13137 2200 ENDIF
36 %13137 2210 For_display(Index_us1)=Prog_code_proc(Index_prog)
22 %13203 2220 Index_us1=Index_us1-1;Index_prog=Index_prog+1
1 %13231 2230 GOTO 2150
0 %13232 2240 ! on to display
20 %13232 2250 Home clear: PRINT '27"h" '27"J"
76 %13256 2260 IF Line_printer$="y" THEN COPY ALL OUTPUT TO "*testli
0 %13372 2270 ! list output by line with detail
52 %13372 2280 IF Line_detail$="n" THEN 2540
26 %13456 2290 ASSIGN #1 TO "copyf",STATUS=Open_err
10 %13510 2300 IF Open_err THEN
37 %13522 2310 PRINT "unable to open copyf, error: ";Open_err
1 %13567 2320 GOTO 2820
0 %13570 2330 ENDIF
4 %13570 2340 ON END #1 GOTO 2530

```

```

0 %13574 2350 IMAGE 40A/40A/40A
104 %13574 2360 PRINT USING 2350;Dashed_line$,"code size analysis by
      Dashed_line$
0 %13744 2370 IMAGE "words"2X"offset"2X"program logic"
46 %13744 2380 PRINT USING 2370
8 %14022 2390 Record_num=1;Index=1
33 %14032 2400 LINPUT #1,Record_num;Input_buf$[1]
231 %14073 2410 IF Input_buf$[11,20]="test probe" OR VAL(Input_buf$[1
0 %14442 2420 IMAGE 4D,X"%5D,68A
214 %14442 2430 PRINT USING 2420;Data_array(Index,4),Data_array(Index
      68]
11 %14770 2440 Record_num=Record_num+1
144 %15003 2450 IF Input_buf$[LEN(RTRIM$(Input_buf$));1]="&" THEN
43 %15223 2460 LINPUT #1,Record_num;Input_buf$[1]
0 %15276 2470 IMAGE 11X,68A
132 %15276 2480 PRINT USING 2470;Input_buf$[1,68]
1 %15502 2490 GOTO 2440
0 %15503 2500 ENDIF
11 %15503 2510 Index=Index+1
4 %15516 2520 GOTO 2400
14 %15522 2530 ASSIGN * TO #1
0 %15540 2540 ! list procedures by size
47 %15540 2550 IF Segment_detail$="n" THEN 2820
102 %15617 2560 PRINT USING 2350;Dashed_line$,"code size analysis by
      subunit",Dashed_line$
0 %15765 2570 IMAGE "segment name"17X"total code"7X"logic code"4X"o
      procedure name"12X"generated"8X"generated"5X"generate
48 %15765 2580 PRINT USING 2570
0 %16045 2590 IMAGE 3X"OUTER BLOCK",15X5D,2X"words"/
59 %16045 2600 PRINT USING 2590;Outer_block_code
44 %16140 2610 PRINT Proc_list$(Num_proc)[15,22] ! output the fir
2 %16214 2620 Total_code=0
18 %16216 2630 FOR J=Num_proc TO 2 STEP -1
0 %16240 2640 IMAGE 3X14A,12X5D,2X"words",X9D2X"words",X9D2X"wor
57 %16240 2650 Dif=Us1_code_proc(J)-For display(J)
160 %16331 2660 PRINT USING 2640;Proc_list$(J),Us1_code_proc(J),Fo
25 %16571 2670 Total_code=Total_code+Us1_code_proc(J)
0 %16622 2680 IMAGE 29X5D,2X"words/segment"/8A
243 %16622 2690 IF Proc_list$(J)[15,22]<>Proc_list$(J-1)[15,22] TH
178 %17205 2700 PRINT USING 2680;Total_code,Proc_list$(J-1)[15,
2 %17467 2710 Total_code=0
0 %17471 2720 ENDIF
33 %17471 2730 NEXT J
5 %17532 2740 Total_code=Total_code+Us1_code_proc(1)
11 %17537 2750 Dif=Us1_code_proc(1)-For display(1)
0 %17552 2760 IMAGE 3X14A,12X5D,2X"words",X9D2X"words",X9D2X"words"
112 %17552 2770 PRINT USING 2640;Proc_list$(1),Us1_code_proc(1),For_d
0 %17732 2780 IMAGE 29X5D,2X"words/segment"
61 %17732 2790 PRINT USING 2780;Total_code
0 %20027 2800 IMAGE /4X"Total code generated: ",8D,2X"words plus st
59 %20027 2810 PRINT USING 2800;Cum_off_set

```

```

8 %20122 2820 Cleanup: CALL Cleanup
67 %20132 2830 IF Line_printer$="y" THEN COPY ALL OUTPUT TO DISPLAY
1 %20235 2840 END
0 %120 2850 SUB Octal to decimal(INTEGER Parameter)
0 %120 2860 COPTION SEGMENT="subseg"
0 %120 2870 INTEGER Final_decimal
30 %120 2880 Final_decimal=0;Local_string$=VAL$(Parameter)
41 %156 2890 Len_of_string=LEN(TRIM$(Local_string$))
23 %227 2900 FOR I=1 TO Len_of_string
155 %256 2910     Final_decimal=Final_decimal*8+VAL(Local_string$
42 %511 2920     NEXT I
7 %563 2930     Parameter=Final_decimal
1 %572 2940 SUBEND
0 %105 2950 SUB Reset_script
19 %105 2960 SYSTEM "file scr=script;save"
19 %130 2970 SYSTEM "purge *scr"
5 %153 2980 SUBEND
0 %107 2990 SUB Cleanup
30 %107 3000 PURGE "sourcef",STATUS=Purge_err
33 %145 3010 PURGE "savecomp",STATUS=Purge_err
30 %206 3020 PURGE "saveusl",STATUS=Purge_err
34 %244 3030 PURGE "copyf",STATUS=Purge_err
30 %306 3040 PURGE "myusl",STATUS=Purge_err
32 %344 3050 PURGE "script",STATUS=Purge_err
28 %404 3060 PURGE "putprobe",STATUS=Purge_err
1 %440 3070 SUBEND
0 %105 3080 SUB Handle_compiler_error(Err_from_mpe,Return_comment
12 %105 3090     IF Err_from_mpe>32767 THEN
22 %121 3100         Return_comment$="fatal error during compile occ
10 %147 3110         GOTO 3140
0 %161 3120     ENDIF
22 %161 3130     Return_comment$="compiler warnings occurred during
1 %207 3140 SUBEND
0 %111 3150 SUB Decide_to_probe(Input_buf$,Yes_or_no)
0 %111 3160 ! is it a sub ??
24 %111 3170     Array_index=POS(Input_buf$," SUB ")
11 %141 3180     IF Array_index=0 THEN 3240
239 %154 3190     IF POS(Input_buf$[Array_index-3;10],"GET SUB") OR
Array_index-3;10],"DEL SUB") THEN 3240
115 %533 3200     Num_value=NUM(Input_buf$[Array_index+5;1])
27 %716 3210     IF Num_value<65 OR Num_value>90 THEN 3280
222 %751 3220     IF LEN(TRIM$(Input_buf$[9,Array_index])) AND NOT P
Array_index],":") THEN 3280
1 %1307 3230     GOTO 3300
0 %1310 3240 ! is it a defined multiline function ??
23 %1310 3250     Array_index=POS(Input_buf$,"FN")+2
113 %1337 3260     Num_value=NUM(Input_buf$[Array_index+2;1])
106 %1520 3270     IF POS(Input_buf$," DEF ") AND POS(Input_buf$," FN
Input_buf$,"=") AND (Num_value>64 AND Num_value<91) T
9 %1672 3280     Yes_or_no=0
1 %1703 3290     GOTO 3310

```

9 %1704 3300 Yes_or_no=1
 10 %1715 3310 SUBEND

 code size analysis by procedure/subunit

| segment name procedure name | total code generated | logic code generated | overhead code generated |
|--------------------------------|-------------------------------|-------------------------|----------------------------|
| OUTER BLOCK | 61 words | | |
| MAINSEG | | | |
| NTEST | 9333 words | 8097 words | 1236 words |
| | 9333 words/segment | | |
| SUBSEG | | | |
| OCTAL_TO_DECIM | 404 words | 299 words | 105 words |
| RESET_SCRIPT | 151 words | 43 words | 108 words |
| CLEANUP | 350 words | 218 words | 132 words |
| HANDLE_COMPILE | 203 words | 67 words | 136 words |
| DECIDE_TO_PROB | 1038 words | 910 words | 128 words |
| | 2146 words/segment | | |
| Total code generated: | 11540 words plus stt code | | |

6.0 CONCLUSION

HP Business BASIC provides the easiest and most flexible access to process handling. Its features provide a wealth of options for the use of process handling in user applications. This new set of options must be accompanied by a note of caution. Business BASIC is the epitome of user friendliness. The joke in the lab is that if you want to hang yourself, Business BASIC will hand you the rope! The point is that the use of process handling, just as any other feature on the HP3000, must be well planned to avoid unnecessary hanging.

I have observed very lethargic response in a process tree scenario used in a well known manufacturing application after installation on a shared processor. The reason for this poor response was that the driving process was a session logged on in the CS queue. It was competing with an equal number of accounting sessions also logged on in the CS queue. The problem was that each manufacturing user was a child process for the driving session and as such was assigned a much lower priority in the processor queue than the accounting sessions. That resulted in the accounting sessions receiving good response times while the manufacturing users received very poor response times. This was resolved by moving the manufacturing application to a dedicated system.

The same caution is in effect for Business BASIC. It is a many featured, powerful application language. Applied properly it can compete with or outperform any of the other HP based languages. Used improperly, it could execute slowly or generate much unneeded code.

7.0 REFERENCES

HP Business BASIC Programmers Guide Part Number 32115-90007

This manual provides a new HPBB programmer with a structured approach to the use of HP Business BASIC.

HP Business BASIC Language Reference Manual Part Number 32115-90006

This manual describes all the HP Business BASIC features with a description of their function, syntax, and options.

HP3000 System Intrinsic Manual Part Number 30000-90010

This manual lists all the system intrinsic available, complete with the parameters, data types, functional description, and syntax.

ACKNOWLEDGEMENTS:

Interpreter - Stewart Hill

Compiler - Jim Preston

Library - John Kwan

System Performance Tuning
and
Memory Availability
HP 3000 Systems

An Investigation Into the Effects of Memory Availability
on Performance for HP3000 Series 48 and 68 Systems

Performed By:

James F. Dowling

For:

Bose Corporation, Framingham, Massachusetts

In cooperation With and
Supported By:

Volz Associates
Winthrop, Massachusetts

and

EMC Corporation
Natick, Massachusetts

November 1, 1985

Abstract

A study was conducted to determine the perceived and measurable effects of main memory availability on HP 3000 Series 48 and 68 computer systems. The objective of the study was to determine the parameters that can be measured using performance monitoring tools that will help to establish the need for and effects of increasing main memory. This objective was tightly coupled to the definition of the term "need" and therefore included a study of "response time" and "throughput" parameters both measurable and perceived. Results and recommendations are presented for system tuning and configuration.

Phase I: Survey

Study Overview

Bose Corporation operates two HP3000 computer systems: a Series-68 and a Series-48. When the need for additional session handling capability was being addressed, several options were included in the study. The Series-48 system could be upgraded to a Series-58 or Series-68. Additional memory could be added to either or both systems. Disc I/O capacity could be improved, or a third machine could be added. This precipitated a need for a global performance analysis. That is: a view into the operation of the current machining hardware and Operating System software to determine where the resource limitation was located. Following describes the process for the Series 68 system. The machine was configured with 3.0 MB of main memory.

To begin the study it was necessary to document the outward symptoms of the problem. Terminal response time was an obvious starting point. By using first-hand observations of various types of programs ranging from graphics preparation to inventory control to on-line inquiries, it was determined that some programs were generally slow, some were slow at times and yet others were seemingly unaffected at all. This is of course the case for most any system and was no surprise. By using several performance monitoring programs system activities were monitored to obtain a quantitative analysis of current activities (Refer to Appendix B for details of the measurable parameters). Overall performance could be described as "sometimes poor but generally good".

The benchmark data indicated that several factors could be contributing to the undesirable performance that was being experienced. CPU utilization was greater than 90% most of the time: ICS (Interrupt Control Stack CPU consumption was in excess of 10% most of the time and MAM (Memory Access Manager) disc I/O was apparently high. The memory manager Clock Cycle rate averaged 0.5 per second with peaks at 1.2 per second. The first three indicators could be interpreted as resulting from inadequate main memory for the system to work with while the fourth indicated a definite problem locating memory. A look at the disc management systems showed that more than 55% of the user disc I/O requests were being satisfied from caching domains and that the average physical disc access rate was less than 30 per second.

Since the program mix on the Series 68 system was fairly consistent from one day to the next, it was assumed that recording relative CPU consumption activity for various processes would be useful to determine how performance problems move around as various resource parameters were altered. To do this several commonly used programs were selected and classified as Transaction Processors, Report Generators, Number Crunchers or Batch Updates. Note that these definitions were for convenience and in some cases (See Appendix B) a reclassification was done later. In general, the system made more CPU available to the Report Generators and the Number Crunchers while the Transaction Processors demonstrated erratic response characteristics and the Batch Updates were rarely serviced.

To determine how much of the apparent CPU shortages were resulting from Disc Caching activity we shut down the caching subsystem and took another set of measurements. CPU utilization was reduced to 75% with some periods at 85%, ICS consumption was reduced to less than 8%, and MAM disc I/O was cut in half. The disc management system overall I/O rate rose from 30 per second to 55.

Under these conditions, the perceived performance of the system was altered dramatically. The most obvious change was a significant reduction in run time for batch processes that were run during heavy on-line activity periods. Most transaction oriented processes suffered increased response time while CPU-intensive operation such as graphics preparation seemed to be unaffected by the change. Report Generation programs demonstrated erratic response characteristics similar to those that the Transaction software suffered in the previous configuration. Overall performance was "erratic and unacceptable, with Disc Caching disabled. It was obvious that some amount of Disc Caching was desirable and that the CPU/Memory issues would have to be dealt with to compensate.

Several attempts were made to use Scheduling Queues and Quantum to effect the desired performance distribution. Results ranged from losing control of the system altogether to processes that were only serviced during lunch breaks. The effects were unpredictable when making radical changes and were insignificant when altered only slightly. We settled on a rather long CS priority queue size and a relatively short and overlapping DS queue. The DS queue overlap was necessary to ensure that Batch processes could have a chance at the processor. It was necessary to limit the number of Batch processes to two during on-line processing periods to avoid batch domination. The CS quantum was set to 100 ms and the DS quantum was set to 300 ms.

Phase III: Memory Availability Testing

EMC Corporation provided us with sufficient memory boards to bring our systems up to eight Megabytes so that we could step the system through various memory size configurations from 2.0 MB to 8.0 MB measuring performance characteristics and resource consumption as we proceeded. For the following tests the general procedure as described above was used. For each test the only parameter that was altered was memory size and data was averaged over fifteen minute periods from 06:00 to 18:00 daily.

At 2.0 mb performance was unacceptable. CPU utilization was at 100%, ICS Overhead at 15-20%, Memory Free Space was located 20-30% of the time, Garbage Collection at 8-11%, MAM I/O at 60-80 per second, and Clock Cycles at 1.2-1.4 per second. Disc caching eliminated 30-40% of user requests.

At 3.0 MB performance was "sometimes poor but generally good", CPU utilization was at 85-92%, ICS at 8-11%, Free Space was located 40-70% of the time, Garbage collection at 3-4%, MAM I/O at 35-42 per second, and Clock Cycles at 0.4-0.8 per second. Disc caching eliminated 45-56% of user requests.

At 4.0 MB: performance was "sometimes same but generally quite good", CPU utilization was at 85-100%, ICS at 4-8%, Memory Free Space was located 70-80% of the time, Garbage Collection at 1-3%, MAM I/O at 30-40 per second, and Clock Cycles at 0.0 to 0.1 per second. Disc caching eliminated 45-60% of user requests.

At 5.0 MB, performance was "sometimes same but generally superior", CPU utilization was at 85-100%, ICS at 4-8%, Memory Free Space found at 85-90%, Garbage Collection at 0-1%, MAM I/O at 30-40 per second, and Clock Cycles at 0.0 to 0.1 per second. Disc caching eliminated 46-60% of user requests.

Conclusions

The primary indicators of efficient performance are "Clock Cycles", "ICS Overhead" and "User Disc I/O Requests Eliminated". These indicators can be interpreted as follows:

Clock Cycles in excess of 0.4 indicate that the Memory Manager is having difficulty locating regions of memory that can be offered up to satisfy a more urgent demand. This results in excessive use of CPU cycles for memory housekeeping functions. More memory will increase the probability of locating an available region without searching. It is important to consider that one clock cycle on an 8.0 MB takes much more work than on a 1.0 MB machine. At the same time, the effort required is proportional to the number of segments in memory rather than memory size. Suggested values are 0.2-0.4 for 5.0-8.0 MB systems and 0.3-0.5 for 2.0-4.0 MB systems.

ICS Overhead is not a particularly good indicator of memory shortage situations unless other indicators are available to support it. Such activities as terminal handling and non-specific activities can cause this number to vary unexplainably. A value in excess of 8% appears to be a breakpoint after which perceived performance degradation occurs.

User Disc Requests that can be eliminated by Disc Caching will convert I/O limitations into Memory/CPU limitations. Increased physical I/O elimination will proportionally increase perceived performance. Caching performance is directly related to the amount of available memory and CPU.

A secondary indicator of memory manager performance is the amount of time spent performing Garbage Collection. When the CPU allocates more than 3-4% of its resource to this operation: the memory manager is having problems finding large enough regions in memory. Increased memory size will help this also.

The HP3000 systems investigated are essentially CPU bound even though they perform I/O intensive tasks. The erratic performance characteristics are attributable to CPU availability. Disc caching has moved the disc I/O bottleneck to the CPU/Memory domain. Anything that can be done to make more CPU available to user processes and Disc Caching will reduce the limitations of two of the system's most precious resources.

Based on these measurements the performance improvement paths that should be considered must include memory additions as a first step before looking into Series 48 to Series 58 or Series 68 CPU swaps. Depending on other constraints such as I/O capacity, significant performance improvements can be obtained by merely increasing main memory to free more CPU for useful work. A thorough analysis of CPU consumption will indicate the potential improvements of increasing main memory.

Additional Observations

During the measurements at 4.0 through 8.0 MB a symptom of sluggishness or intermittent poor response was observed. By making measurements of various processes to determine CPU distribution it could be seen that the priority of seldom active processes was significantly higher (high priority processes are dispatched after those at a lower priority) than those that operated frequently. Moreover, the priority of these processes rose rapidly while some few processes always managed to work their way back to low priority after being rescheduled.

An investigation of these processes showed that the Dispatcher/Scheduler prioritization scheme was working perfectly albeit against our desires. Of particular distress was the urgency with which graphics output programs were put at the head of the list only to drop to the end a moment later then get pushed back again and again. The result is a bizarre effect where the process consumes a disproportionate share of CPU yet takes more elapsed time to complete.

Another manifestation of this sluggishness is experienced when many interactive processes quiece (user think time) and a few CPU-intensive processes are running. The effect is to have a set of processes that do not fall in priority rapidly enough to get the interactive processes running quickly. The symptom at the user terminal is that of a dead system for a few seconds.

One effect that was looked for but not found was that of a system that might squander main memory if it were available using memory scanning programs it was found that when large memory configurations were being used by small numbers of users there were large areas of unused memory in the higher banks. As the user count increased these banks became populated then released as the user count dropped off. This is a good sign that Disc Caching does not complicate the Memory Manager's problem by overpopulating memory with cache domains that would later require removal.

MPE changes can move performance constraints from one resource to another, making it difficult to provide a lasting solution. MPE V-E appears to require more CPU resource for its own purposes as well as that consumed by user processes than did MPE V-P. It also requires more memory just to get going. From one release to the next we have seen a consistent increase in the amount of stack space required by the file system intrinsics. Disc Caching takes advantage of reduced memory costs to improve the disc I/O situation as will TurboIMAGE. All of these performance and feature enhancements have relied on sufficient memory and CPU availability for best results. At best, inadequate memory will cancel the positive effects of a software improvement. At worst, it can cause a performance degradation.



Reference Material

D.Beasley: "How Dispatching Queues Really Work": Proceedings of the INTEREX Conference: Washington, September, 8, 1985.

D. P. Beauchemin, "Things that Go Bump in The HP3000," Proceedings of the INTEREX Conference, Washington, September 8, 1985.

J.R. Busch, "The MPE IV Kernel: History, Structure, and Strategies: Proceedings of the HP3000 International Users Group Conference, Orlando, April 27, 1982.

J.R. Busch and A.J. Kondoff, "MPE Disc Cache: In Perspective," Proceedings of the HP3000 International Users Group Conference, Edingburgh, October 1, 1983.

B.Carroll, "MPE Disc Caching," Proceedings of the INTEREX Conference, Washington, September 8, 1985.

B. Duncomb, "Performance Self-Analysis," Proceedings of the INTEREX Conference, Washington, September 8, 1985.

E.Volokh, "The Secrets of System Tables...Revealed!" Proceedings of the INTEREX Conference, Washington, September 8, 1985.

APS/3000 Users Manual, Hewlett Packard Co.

MPE V-E System Tables Manual, Hewlett Packard Co.

OPT/3000 Users Manual, Hewlett Packard Co.

SYSVIEW Manual, Second Edition, Carolian Systems, Inc, September, 1984.

Appendix A

Performance Measurement Parameters

Following are descriptions and interpretations of many MPE and hardware operation parameters. They are described here to help use the data presented in this paper as well as when using the various performance measurement software tools. The descriptions are therefore not necessarily classical definitions, but rather, they are useful interpretations.

Background Garbage Collection: When the memory manager determines that is under memory pressure it will perform some memory housekeeping to make larger regions available. This is done when the system would otherwise be idle. Excessive CPU activity here indicates a shortage of main memory. See also Garbage Collection and Memory Pressure.

Clock Cycle: The memory access manager maintains a pointer into main memory as a reminder of where to start looking next time it needs a region of memory. As it proceeds through memory, it pushes the pointer along ahead of it. When the pointer passes the start of main memory one clock cycle is said to have passed. The number of clock cycles per second indicates the current level of Memory Pressure.

Code Segments: These are a special case of Data Segments that are used to store executable machine instruction sequences. Code segments can be shared by many processes.

Data Segment: The MPE operating environment consists of Code Segments and Data Segments. The distinction is semantic as far as memory is concerned but convenient when discussing system operation. Data segments generally contain only process operation data or control tables. With rare exception, Data Segments are not shared by multiple processes.

DEVREC: When an Interrupt is received from an unexpected source such as a magnetic tap coming on line or a terminal character, this process determines how to handle it.

Disc Caching: To help the HP3000 systems overcome a severe shortage of I/O throughput a mechanism for storing disc images in main memory was introduced. This enables the file system to resolve some percentage of the total disc requests from memory thereby improving performance. This use of CPU and Main Memory to enhance disc performance is called Disc Caching.

Extra Data Segments: Each process is associated with one data segment, called its stack. It may obtain additional memory areas for data called Extra Data Segments.

Garbage Collection: When the memory manager finds two adjacent regions in memory that are both available, it will combine the two into a single region. This activity is generally called Garbage Collection. A certain amount of garbage collection will always occur. Therefore, this is not a good memory pressure indicator. See also Background Garbage Collection.

Global Performance Analysis: This refers to the process of pressuring the performance of the system as a whole by determining how the system resources are being allocated among the various potential consumers. See also Local and Specific Performance Analysis.

Interrupt Control Stack: MPE is an interrupt driven Operating System. A device or process that requires attention places a request onto the Interrupt Control Stack. The Dispatcher/Scheduler and many Interrupt Handlers such as DEVREC use the ICS as their Stack. CPU time spent on the ICS operations is done as a result of user process requests but is not considered useful work.

Local Performance Analysis: This refers to the process of determining which of the contending resource consumers is obtaining or not obtaining its share of the available resources. The Process Context of OPT/3000 or SYSVIEW would be used for this. See also Global and Specific Performance Analysis.

Memory Pressure: When the memory access manager cannot locate an available region in memory for a data or code segment, it must perform some housekeeping work to alleviate the situation. The amount of work required is called the level of memory pressure.

Memory Access Manager: (also known as MAM, MM and Memory Manager.) This process manages all memory allocations and deallocations for MPE. MAM uses two forms of Virtual Memory to handle memory shortage situations and memory region changes. See also Virtual Memory. Time spent on memory allocations is to be expected and should be somewhat related to user count. If little time is spent here, ample memory must be available.

When MAM alters a data stack size or reorganizes memory it will swap the affected data segments to virtual memory. When additional code segments are required for a process or swapped data segments are required, they will be read from disc. Excessive MAM disc I/O can indicate a memory shortage.

MPE: The MultiProgramming Executive is the HP3000 Operating System. It comes in various capability sets and release levels.

- Process:** The unique execution of a program by or in behalf of a system user. Each process is identified by a unique Process Identification Number (PIN).
- Quantum:** This is an archaic term for the maximum amount of CPU that any process will receive before it is rescheduled. Actually a more sophisticated mechanism is used by the MPE-V Operating System to manage maximum CPU utilization and rescheduling.
- Response Time:** For the purposes of this study Response time is defined as the interval between pressing the Return/Enter key and the ability to either modify the retrieved data or to request another retrieval. This therefore includes screen painting and fill as well as the time necessary to obtain access to the processor. This is not measurable by any available software.
- SYSVIEW** defines this (for its purposes) as the time from pressing Return/Enter to the delivery of the first character of the retrieval to the terminal handler. This is a measurable entity.
- Scheduling Queues:** MPE supports five queues for scheduling processes. The AS and BS queues are reserved for system processes. The CS, DS and ES queues are for user processes. Processes within each queue are given access to the CPU in prioritized order A - E and by priority within each queue. See the Reference Material list for more on this.
- SL Files:** Operating System as well as user process code can be stored in disc files which are loaded into memory when needed for execution. The SL files are used to store codes that many processes can share rather than duplicating it for each process.
- Specific Performance Analysis:** This refers to the process of determining why a specific program is either consuming or not getting its share of system resource. This generally requires specific knowledge of the intent and structure of the program. APS/3000 and TINGLER provide the necessary run-time analysis tools but a source code/data base walk through is usually required.
- Stack:** When any MPE process is active it operates on data that is located in a memory segment called a Data Stack. This stack is managed by the user and operating system code. An alternate location for process data is an Extra Data Segment.
- System Tables:** A special set of data segments are set up for use by MPE. These areas are formatted and managed only by MPE.

Virtual Memory: MPE can handle more processes and data areas than main memory would allow by utilizing two types of disc base memory alternatives. Data Segments are written to reserved disc areas called Virtual Memory Domains. These may be spread over several disc drives. Code segments come from SL files or PROG files. These files themselves provide the Virtual memory space.

Appendix B

Classification of Processes

The classification process was used to categorize several processes according to their balance of resource consumption. Disc I/O and CPU intensity determined the classification as follows:

Transaction Processors impose relatively small I/O and CPU loads on the machine at random but widely spaced intervals. The typical inventory transaction processing application is in this class.

Report Generators impose heavy I/O then CPU then I/O loads on the machine at widely spread random intervals but the load duration is quite long. Utility report writers are in this class.

Number Crunchers impose heavy CPU loads for long durations with relatively little I/O required. Graphics and Spreadsheet processors are in this class.

Batch Updates impose a peculiar load on the system. They can be looked at as transaction processors with no wait time between transaction. The more serious difficulty is that they seize resources such as files and data bases but are scheduled as very low priorities. This causes them to put a continuous load on the machine as well as tying up other potential accessors to the file or data base.

Some processes were reclassified after the initial review to better accommodate their characteristics as displayed with Disc Caching disabled. The reclassification was useful so that Scheduling Queue, Quantum and Disc Caching experiments could be compared directly.

Appendix C

System Tuning

CPU, Memory and Disc I/O make up the primary resources that the System Manager must provide and apportion to the system users. The degree to which they are available and fairly distributed determines the level of perceived system performance. In most cases tuning is not an attempt to optimize the use of these resources, rather it is an effort to cause them to be made available uniformly across a large user population. A clear understanding of how MPE manages these resources is essential if one is to successfully establish a set of operating parameters for MPE to work with. Following are the resources and user-alterable parameters that are available in a Global context:

Resources:

CPU- From the Series 37 through the Series 68, many levels of central processor capability are available. For any given processor level, the system manager should attempt to minimize overhead activities such as memory management and communications software so that more of the CPU will be available for user processes. RJE/3000, MTS/3000, IML/3000 and DS/3000 all consume significant CPU but no alternative other than elimination exists for controlling their consumption rates. Disc Caching consumes CPU in exchange for reducing user process wait time for Disc I/O. This is a good area for controlling CPU consumption by trading caching advantages for CPU availability. Another, less obvious CPU consumer is the process loader program. It can only load one process at a time and when in action it operates at a high priority.

MEMORY - System Tables and Disc Cache domains are the only manageable parameters. Use particular caution when attempting to keep System Table Sizes small because several can only be changed at a Reload and lack of available space in others will cause a system failure. Data presented above indicates that disc caching will surrender memory if user processes need it. Therefore the system manager should ensure that sufficient memory is available for all contenders then let MPE handle this resource.

Disc I/O - Even with disc caching this is a precious resource on any HP3000 system. At the global level the primary consumers of disc transfers are the memory manager, disc caching and the process loader. Memory management requires disc activity for virtually (pun intended) every memory allocation. Each data segment that is allocated in memory also has an area of disc set up to handle the eventuality that the memory space may be needed for another purpose. If so, the memory image is copied to disc then brought back later when needed. By keeping memory availability high, these swaps will be minimized. Disc caching performs disc I/O in excess of that requested by the process in the hope that subsequent accesses will be satisfied without requiring another physical disc access. Several parameters control this activity and will be presented in detail below. It suffices to say that the more memory that is available the better the caching advantage. Control should be exercised to get the most out of the additional data that is transferred. Code from PROG and SL files is loaded from disc whenever needed. It will be removed from memory only if necessary to fulfill a higher urgency need. By "allocating" programs and SL segments, much of this high priority disc activity can be eliminated.

Tuning Parameters:

CPU activity can be controlled both directly and indirectly by using two sets of tuning parameters. The scheduling parameters available through the TUNE command directly effect the process dispatch/scheduling algorithm while the disc caching parameters available through the CACHECONTROL commands effect the extent to which disc caching converts I/O limitations into CPU consumption. When a system is supporting large user populations the balance of these two parameters seems to have the most profound effect on perceived system performance as follows:

It appears that there are two types of transactions that evoke different expectations. One type, such as a simple Customer Inquiry should occur quickly all of the time whereas a Customer listing or converting a vector drawing to a raster image need not complete so quickly. Essentially, the user imposes a subjective performance expectation based on his perceived level of difficulty. When the memory and disc I/O limitations are reduced through disc caching and providing sufficient memory, the schedule/dispatcher determines the actual performance of both transaction types. The irony is that when there is significant CPU demand, just the opposite effect is produced. The multiple data set search for a small amount of data gets snarled by the CPU-intensive calculation and the ever-ready on-line data retrieval from a single file. This is an explanation of the term "Sluggish" as used above.

The solution to this involves several coordinated steps which are intended to more fairly distribute the CPU:

Set the CS and DS scheduling queues so that there is no overlap and to provide a broad distribution of CS (interactive) processes. The default parameters for MPE-V are CS = 152 to 200 and DS = 202 to 238. Using CS = 152 to 200 and DS = 222 to 238 will provide the additional spread and maintain the necessary overlap to prevent Batch Job encroachment on interactive processes.

Set the limits for the Average Short Transaction Time to compensate for the minute transaction length of such processes as listing data to terminals and plotting. In both cases, the "are you ready for more" inquiries to the terminal, printer or plotter dominate the AST calculation causing all other processes to be considered CPU jogs and therefore penalizing them. If this is the case on your system, try setting CS MIN and MAX to 50 and 350 respectively. For the DS queue no variations from standard settings will generally be necessary.

Use the ALLOCATE command to set up the external references, Code Segment Table references and Extended Code Segment Table reference for your most often executed programs. Caution must be used to avoid overflowing the CST, DST and XCST tables. To allocate a program that uses SL segments, you can RUN it on one terminal then ALLOCATE it from another.

Once the system is running with a typically heavy user load, the parameters of disc caching system can be monitored and adjusted. The method will be to optimize the performance gains for the caching system by adjusting Sequential and Random fetch quantum then to adjust the caching activities to reduce CPU load if necessary. The SHOWCACHE command will give you all of the information that you need to determine the effect of parameter changes. The key value is % of user I/Os eliminated.

Our first objective will be to maximize caching effectiveness. With sufficient CPU and memory numbers from 50 to 70% should be achievable for % user I/Os eliminated. The sequential fetch quantum default is 96 sectors or 24 Kbytes per disc request. Adjusting this number downward will increase the number of physical disc accesses if the majority of the processes are actually processing the files from one end to the other contiguously. Conversely, increasing the quantum should reduce the number for such processes. By altering this setting from the default and monitoring the effect, an optimum setting can be located. Bold changes on the order of 25% can be used to quickly locate the thresholds both minimum and maximum.

The random fetch quantum on the other hand should be dealt with a bit more precisely. The default of 16 sectors is tuned to the buffer size of the CS80 disc drives. A larger quantum will require additional disc wait time and therefore may degrade performance if only a small increase is attempted but a larger change may then show improved performance. Note that all IMAGE/3000 disc I/O is random as is most KSAM/3000. Note that caching parameters can take several minutes to take full effect.

Having established desirable settings for fetch quanta, the next global parameter to work with is Block on Write. This will effect process scheduling as well as determining the urgency of some disc activity. If your system is typical, it will perform significantly more reads than writes. SHOWCACHE will give you a good handle on your systems read/write ratio. A Read % figure of 80% indicates a 4 to 1 read/write ratio and therefore implies that if you want to free up some CPU by penalizing processes that perform a lot of disc write activity, you can simply set blockonwrite to YES. This is a good idea for several other reasons including file system integrity.

If the system is still demonstrating erratic response characteristics or if CPU appears to be unavailable (excessively long logons or batch jobs that run forever), you can trade some of your disc caching advantage for more available CPU. Again, using the SHOWCACHE command, locate a disc that demonstrates the least caching performance as indicated by the total number of cache requests that were satisfied from cache domains.

$\text{Read Hits} = \text{Cache Requests} * (100\% - \text{Read}\%) * (100\% - \text{Read Hit } \%)$

By applying the above formula to each disc, you will compute the effectiveness of caching. Stopping caching on the drive with the lowest number of read hits will return some CPU with the least global reduction of caching advantage. Note that specific programs will be affected more than others if their file complement happened to reside on the chosen disc.

A memory shortage will not generally be amplified by disc caching to any great extent but, in extreme cases it may be worth a try to suspend disc caching on the drive that demonstrates the least caching performance with respect to the amount of memory that is being used. By dividing the Read Hits calculated above for each drive by the number of K-BYTES of memory used, the disc with the least performance can be located. Suspending caching on this drive will give memory back to the system but not necessarily at the lowest global reduction in caching advantage.

The steps outlined above have produced positive results in various environments. Since performance tuning merely moves the resource shortage from one area to another, the cycle should be repeated to determine if a new symptom arises. The scheduling parameter and caching adjustments are particularly sensitive to CPU and memory loading. Therefore user count and process activity will effect the benefits that are gained. It is highly recommended that you keep a good set of notes and after a period of several days to two weeks a median set of operational parameters be selected.

**The Hewlett-Packard Executive (HPE) Data Management System
Hosting the MPE-XI File System Environment
for HP's Next Generation Commercial Computer Systems**

by Alan J. Kondoff

HPE FILE SYSTEM

Three primary objectives needed to be satisfied through the implementation of a new, high-level data management subsystem in HPE. These were:

- Exploit performance advantages available through HPE on the HP Precision architecture.
- Provide a subsystem architecture that efficiently supports the semantic and programmatic requirements of multiple host environments.
- Provide an extensible base that can track the functional and availability evolutionary directions of HPE.

PERFORMANCE ADVANTAGES

Single Level Store

The most significant advantage of the HP Precision architecture, exploited by the data management subsystem, is its extremely large virtual address space and main memories. All accesses to secondary store (disc) is performed via machine load and store instructions to regions (disc files) mapped into the machine's virtual address space. This eliminates the need for explicit disc file buffer management and location. Buffer location is performed by HP Precision hardware, while management functions are handled by standard HPE memory management facilities.

By providing a byte-array abstraction of secondary store to the disc access methods, a simple, deliberate code algorithm can be designed to access each type of disc file. To an access method, a disc file appears as a four gigabyte array of bytes onto which it can impose its specific organizational and semantic rules.

User Mapped File

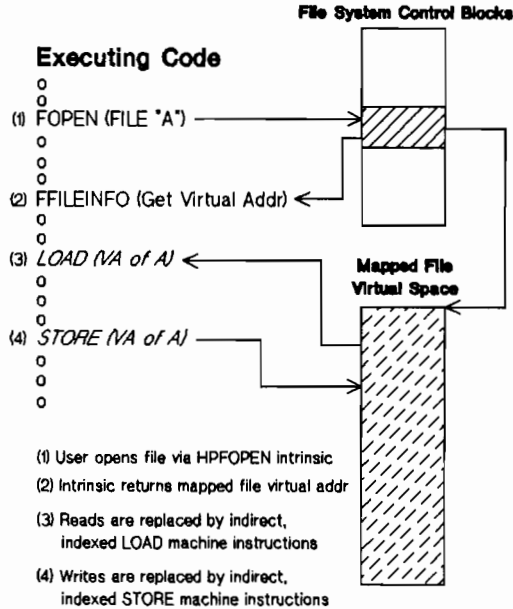


Figure 1 : User Mapped File

A fallout of the single level store nature of disc file access on HPE is the notion of a user mapped file. Certain access methods allow users to directly modify the contents of a disc file without going through the access methods. A virtual address can be passed back to the application (user) through which accesses to the disc file can be performed through a simple, dereferenced pointer. The application now has access to a disc file at HP Precision load/store machine instruction speed without incurring the additional overhead of file system access methods.

Since user mapped files are under control of the HPE File System, all security, protection, and rendezvous mechanisms are defined and enforced as would conventionally accessed files. This allows applications to make use of user mapped files as shared, named common storage between programs, as a basis for their own specific disc access method, or as retained working storage. Mapped files offer applications the opportunity to achieve high levels of performance through conventional programming practices.

Integrated Mechanisms

Key secondary storage management mechanisms have been strategically integrated with disc data management to effect the most optimal system resource utilization possible under varying load conditions.

The behavior and reference pattern of disc accesses has been extensively measured and modeled for a wide range of application mixes on commercial HP systems. Through these analysis and experiences with internal disc caching on MPE-V based computer system, a set of relevant metrics and mechanisms emerged. Disc prefetch, posting, extent allocation, and placement were among the mechanisms selected for integration.

When a page fault against a disc file is detected or projected to occur on HPE, a strategy routine is invoked to determine how much data should actually be brought in, or prefetched, from disc. The goal of this strategy routine is to provide enough data to keep the application running without unnecessarily over committing disc or main memory resources. The strategy routine consults and adjusts a set of metrics, against which it makes its decision.

The prefetch strategy routine views the essence of both global and local references against a disc file, and makes a heuristic determination of how much data should be prefetched from disc. The algorithm used is deliberately simple. Disc file metrics consulted indicate file consumption rate, access pattern (sequential or random), fault rate, access method hints, and global main memory availability. Through this localized file view and global feedback metrics from the prefetch mechanism, the file system can adjust to varying application demand, CPU availability, and main memory availability to provide the best global result in a dynamic environment.

The posting (modified data write-through to disc) strategy routine uses the same set of metrics as does the prefetch routine, but to a slightly different set of goals. The posting routines attempt to judiciously manage main memory occupancy, file consistency windows, and disc utilization.

Main memory occupancy of file pages is primarily the responsibility of the memory manager. The file system posting routines assist the memory manager's job in several ways. When sequential access is performed to a file, the posting strategy routine explicitly requests the memory manager to post large, consecutive virtual address ranges. In addition, memory manager is also informed that there is a low probability that these pages will be referenced in the short term, which makes them readily available for replacement if main memory is needed. This input to memory manager eases its ability to claim needed main memory pages on demand while minimizing the number of visits to disc in order to post modified pages (minimizing disc utilization).

The posting strategy routines are also concerned with file consistency. The notion of dependency queues have been implemented in the memory manager to satisfy atomic post and asynchronous post order constraints. File posts (or writes from the user perspective) can be piggybacked on one another without waiting for physical completion to disc. Applications can now post any number of files without waiting and be guaranteed that the order of the posts are maintained. By merely waiting for the last post to physically complete on a dependency queue, the entire chain of requests are guaranteed to be durable on disc. This allows the greatest possible disc throughput while minimizing process stops required for transaction consistency checkpoints.

Disc utilization is optimized through the previously mentioned prefetch and post strategies, along with extent management. When a page fault occurs on a portion of a disc file that does not have disc space allocated, a strategy routine is invoked which will determine the size and placement of the file extent to be created. File extent sizes on HPE are variable, with no practical limit on the number of extents in a file. The metrics consulted when determining the new extent characteristics are dominant access mode (sequential or random), file capacity, access method hints, and extent fault frequency.

Sequentially accessed files will tend to have large extents consecutively located on a disc drive. This allows the prefetch and post strategy routines to minimize visits to the disc, increasing effective disc utilization. Random access files will tend to have smaller extents spread across disc drives, allowing highly parallel access to data for both prefetching and posting. Extent allocation size may also be



modified due to actual remaining disc space on a drive, allowing full (100%) utilization of the media by the file system.

Specialized Mechanisms

The throughput of the data management subsystem has been enhanced through the implementation of specialized mechanisms in HPE. These mechanisms address systematic problems in commercial computing systems. The areas can be loosely categorized into increased data concurrency and repetitive disc file reference.

HPE has integrated a full functioned transaction management facility within its disc data management subsystem. This facility allows all users of HPE data management services to optionally make use of the integrated lock, log, and recovery facilities in transaction management.

All permanent data structures managed by the HPE File System have the transaction management property enabled. This allows highly concurrent access to system directories and disc file labels through shared or exclusive page level locks. HPE transaction management will be discussed later in this article.

Granular use of shared and exclusive semaphores in data management services has virtually eliminated artificial points of serialization in data management. Concurrent file opens, closes, page faults, disc file map-ins to virtual space, and name space resolution can occur. If contention does arise, automatic process priority elevation mechanisms alleviate convoy effects. In many instances, like exclusive file access, all locks are avoided by access methods to give optimal performance.

Repeated reference to disc files are enhanced due to a closed file LRU (least recently used) list. All files that are closed (no users) on HPE are placed on LRU and remain mapped into virtual space with file pages remaining in main memory. When an initial file open operation locates a file on the LRU, it is merely removed from the LRU and reactivated. The file open operation and subsequent accesses can potentially occur with no disc accesses. This is especially significant for operating system functions like directory scans, or job steps in user applications where the output of one program feeds the next.

Temporary or new files are also treated specially in HPE. By virtue of their transient nature, the HPE file system will avoid posting data to disc unless main memory is required. It is possible for a temporary file in HPE to be created, accessed, and deleted without a single disc file access.

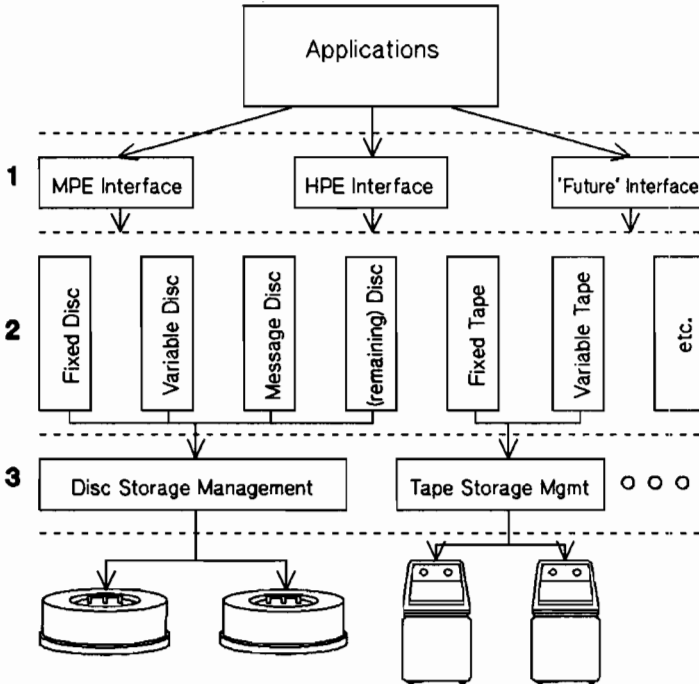
Accessing new portions of new disc files, or extending the end-of-file on existing files, also has special optimizations. When a page fault is detected on one of these previously untouched or virgin pages while accessing a file, a disc transfer to read the page is avoided. Memory manager merely claims available page(s) in memory and initializes it to the file's fill character (usually blanks or zeros).

FILE SYSTEM ARCHITECTURE

Overview

The organization of the HPE File System has been optimized to efficiently support multiple external operating system environments via a common set of file system services. Three levels constitute the primary access method path in the file system, and are called the storage management, type management, and intrinsic interface layers.

Structure



- 1-Intrinsic Interface Layer (Environment Dependent)
- 2-Type Manager Layer (HPE basic services)
- 3-Storage Management Layer (HPE basic services)

STRUCTUR

Figure 2 : File System Structure

The highest level of the file system hierarchy is the intrinsic interface. Each intrinsic interface module can efficiently support the procedural interface, semantics, and error conventions documented for a specific host interface. For the first release of HPE on HP Precision, the MPE Intrinsic Interface module will be included in the file system. Future releases of HPE may include additional intrinsic interfaces which may coexist with other intrinsic interfaces in the HPE File System. Host environments can also be interchangeably invoked from user applications.

Type managers occupy the middle level of the file access hierarchy. They define a consistent set of interfaces and operators which may be applied to a file through the intrinsic interface. All intrinsic environments access a specific file type through the same type manager, insuring integrity of the file. Access to a particular type manager, or type manager operator, may be restricted by an intrinsic interface, but is not enforced by the type manager. Internally, type managers provide a logical abstraction (fixed, variable, byte stream, keyed, etc) of the specific storage management module being accessed.

Storage management is the lowest layer of the three abstract layers of the HPE File System. It defines the set of basic operators that can be performed against a specific class of devices (discs, tapes, terminals, etc). Within the disc storage management module, additional subsystems such as transaction and disc volume management are also included.

Binding between the three levels is performed at file open time. The file open module determines the proper storage management and type management module from the file label, and performs a one-time binding between the levels for all subsequent accesses. Establishing the access path up front, through specific type and storage management levels, permits HPE data management to execute short, deliberate code sequences.

Namespace Resolution

The HPE File System maintains an internal file name, called a UFID (unique file identifier), for every file in the system. The identifier is a combination of the unique media identification and a timestamp. This UFID provides a unique file handle satisfying both network and transaction management requirements. HPE File System services use this handle as the rendezvous mechanism between users and files.

Intrinsic environments define differing naming conventions, accounting, and security semantics. To support these requirements, the intrinsic interface is responsible to provide a name server module which will convert the external, environment specific name to an internal HPE File System name (UFID). Differing security and accounting semantic support, beyond the base HPE facilities, may also be required. This support is included as part of the namespace module for an intrinsic environment. Any additional security or accounting on a file is thus a function of the namespace that a file resides. For first release of HPE on HP Precision, the MPE namespace module is provided.

TRANSACTION MANAGEMENT

In order to satisfy the HPE goals of extremely high transaction throughput rates, high data availability, and future functional evolution, an integrated transaction manager has been incorporated into the disc storage management subsystem. Through the integration of the transaction manager into HPE and the HP Precision architecture, the most efficient, extensible implementation possible is realized.

The classic approach to providing transaction management functionality in a commercial computing system has been ad-hoc, at best. Concurrency control mechanisms were either provided by file system access methods or database management. Recovery, auditing, and backup/recovery mechanisms were provided by file system, database, and application facilities. To further complicate issues, different file system access methods, databases, and applications tended to manage each mechanism differently. The result was a complex solution, requiring duplication of effort, high administrative and support overhead, and compromised performance for a customer installation.

By consolidating all these functions into a single module, common to all disc access methods, HPE has achieved a consistent and efficient facility that all applications can utilize. Performance and efficiency gains are also realized over implementations of these facilities at higher levels of the system due to tight coupling with memory management, I/O, and HP Precision protection hardware.

The following sections describe the three basic facilities provided by the HPE transaction management facility. These are the recovery manager, log manager, and lock manager.

Lock Manager

The concurrency control mechanism in a system can be viewed as a scheduler. It accepts begin, data access, and commit requests from transactions, and decides whether to allow, postpone, or reject these requests. In order to control concurrent transactions accessing shared data, a concurrency controller has been implemented in the HPE File System.

The unit of locking in HPE is HP Precision page (2048 bytes). Implicit locking (load/store access) locks single pages, while explicit locks by virtual address range may accumulate multiple page locks. When a transaction is granted access to a HP Precision page, a HP Precision protection identifier (PID) is placed on that page which allows only those processes participating in that process to read or read/write that page. Any other process attempting access to that page is trapped by hardware to the lock manager.

The user is provided with two intrinsics to define the boundaries of a transaction. They are: Begintran and EndTran intrinsics. Locks are claimed for file system operations as they are executed. Sometimes, it is referred to as locking on-the-fly. Under this scheme, it is possible for deadlock to occur. A simple example of deadlock involving two transactions is shown in figure 3.

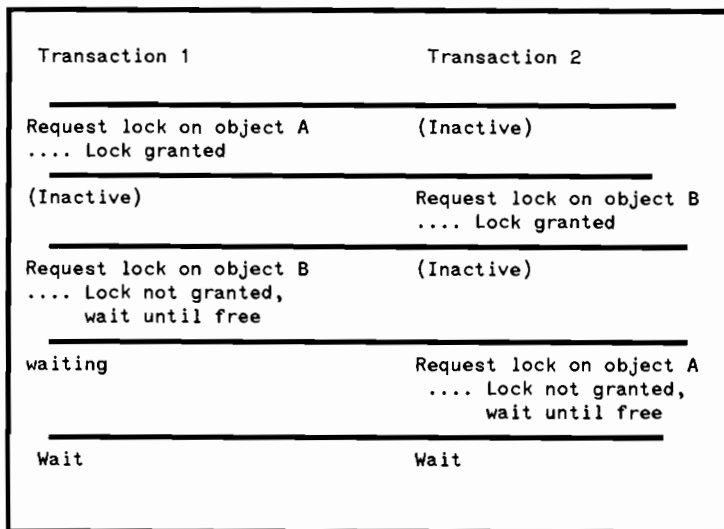


Figure 3 : Deadlock Sequence

Each transaction is waiting for the other to finish and release its locks. The HPE lock manager will detect the deadlock and resolve it by backing out one of the transactions, releasing its locks (thereby allowing the other transaction to proceed), and then notifying the program so that it may perform clean up and optionally restart the transaction. Extensive analysis has shown that deadlock probability is low, and that throughput advantages over classic predicate locking schemes is substantial.

Log Manager

Logging and recovery provide both physical and logical disc file consistency. Physical consistency is provided in order to preserve structures within the files in case of system crash. Logical consistency is provided in case of a user transaction abort, user process abort, system crash, or hard crash.

Logging maintains enough information to preserve consistency at the transaction level and guarantee that the recovery manager to be able to perform soft or hard crash recovery. In order to implement transaction abort, logging records contain enough information about each user operation against a set of files to undo the actions. This information, which includes the content of the before and after image of the modified virtual address range, is written in the recovery log.

The recovery log also contains enough information to handle hard crash. An older version of the file system can be restored, and subsequent logs are used to bring it up to date. During recovery, if a transaction is incomplete or aborted, sufficient data is available in the log to undo the transaction. In other words, a transaction that is incomplete or has been aborted will appear in the recovery log as a NIL transaction.

Application specific information can also be placed in the log for auditing or journaling requirements.

Recovery Manager

The HPE File System provides optional logical and physical disc file recovery, relieving the burden of applications providing their own recovery. Application updates to files that have a recovery property are recorded in a recovery log. A file is said to have a recovery property if it is attached to a recovery log. The recovery log is recorded on disc, and may be duplicated. Typically, there is one recovery log for a set of logically related files or application.

Recovery handles three types of failures:

1. Program transaction aborts occurring at run time. An implicit abort transaction (through program terminations without logical transaction conclusion), or via explicit abort transaction issued by a program.
2. Soft crash failure (hardware or operating system failures). It is assumed that all disc hardware and media is in a good state.
3. Hard crash failure (disc head crash or unrecoverable media).

The HPE file system provides the AbortTran primitive in order to handle program aborts and user transaction aborts.

If a soft crash failure occurs, recovery reads the log in order to restore the work of all committed transactions that are not reflected on disc and the user has seen as committed. It will also abort all uncommitted transactions.

Recovery from media failures requires an archive copy and the recovery log. In order to do recovery from disc head crashes, the file store/restore facility restores the archive copy to disc, followed by the Recovery Manager which reads the recovery log to redo all committed transactions. In order to provide high data availability, the HPE File System supports mirrored disc logs and facilities to extract the logs off marginal media.

BACKUP/RECOVERY

The HPE backup/recovery subsystem supports an enhanced version of the MPE store/restore interface. It implements a number of new options which provide the user with the ability to save files dynamically, utilize multiple backup devices and increase backup throughput using file interleaving techniques.

System downtime is significantly reduced by using the dynamic backup capability in HPE. The integrated transaction management system allows any set of files attached to a log to be backed up while applications are concurrently accessing and modifying the files. Log files are stored along with associated data files allowing recovery to a logically consistent state when the files are restored.

File not under control of transaction management, i.e., not attached to a log, are attached to a log created for the backup. Logging remains in effect until the backup is completed. Again, the files are recovered to a logically consistent state during restore. The limitation for files not under transaction management control is that any file opened for write access at the time of the backup cannot be stored dynamically because recovery to a consistent state cannot be guaranteed.

HPE backup also allows the user to specify a number of performance options which each increase backup throughput to a different degree. Multiple backup devices may be used for a single backup providing concurrent access to the backup media. Multiple file extents may be gathered into a single block and stored in a single I/O request (file interleaving) allowing a higher input rate from secondary storage.

The backup subsystem was constructed around a core module which supports the basic input and output operations. This native mode module is integrated with the virtual, main and secondary storage systems to support the the rates of backup throughput.

A higher level set of modules provide the user interface to the backup subsystem. These routines support an MPE compatible command set as well as the enhanced functions.

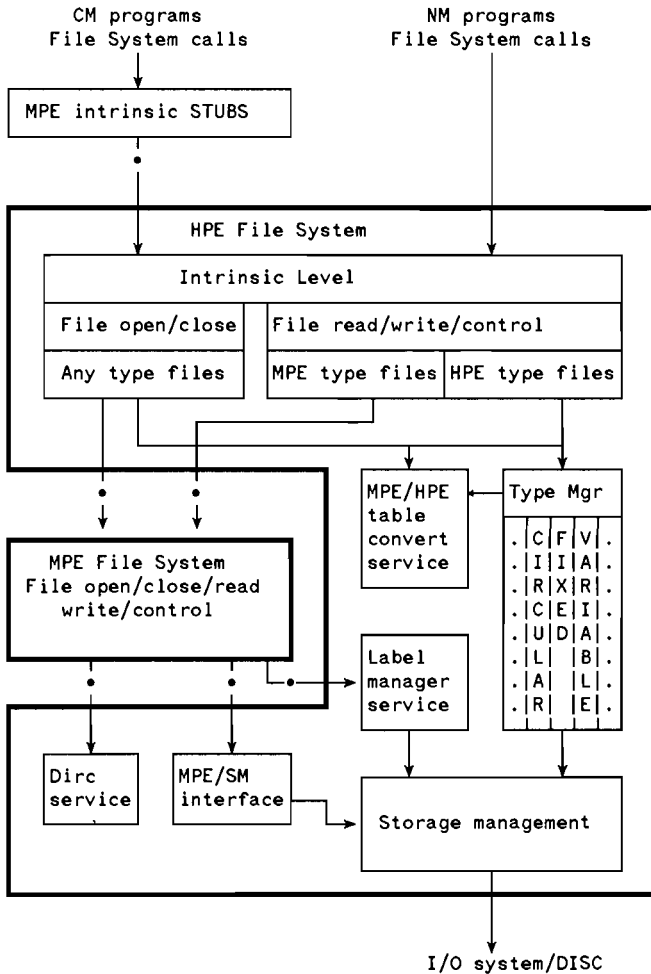
COUPLED ENVIRONMENT

The term coupled environment describes the HPE File System's usage of the MPE-V file system code for low frequency functions thereby providing full MPE-V file system compatibility at first release. This dual (coupled) file system environment was developed to:

1. provide 100% MPE-V file system compatibility (at initial release).
2. provide a migration path for future HPE File System functionality. Migration of additional functionality can be easily performed in subsequent releases by assuming responsibility from the MPE file system.
3. provide full HPE File System performance for paths not relying on the coupled environment. No performance penalty is incurred due to the presence of the coupled environment.

All MPE file system calls are intercepted by the HPE File System's MPE Intrinsic Interface. The appropriate type manager is called, and if it the ported MPE file system type manager, the ported MPE code will be invoked.

Figure 4 describes the flow of an MPE intrinsic in the coupled environment.



- : Indicates the environment switch.
- * : For circular disc file only.

Figure 4 : Coupled Environment

HPE VOLUME MANAGEMENT

**Rick Ehrhart
Hewlett-Packard
19447 Pruneridge Ave.
Building 47UE
Cupertino, CA. USA 95014**

1.0 Introduction

This paper discusses the new Volume Management facility of the HPE operating system. HPE Volume Management handles disc volume sets, volume classes, and volumes. It creates volumes, mounts volumes, and informs the operating system about volume related data. This paper will cover the specifications and the design overview of Volume Management.

2.0 Design Objectives

HPE Volume Management design objectives are:

- **Interface that is consistent with usage**
- **Natural, no performance penalty**
- **Programmatic compatibility with MPE V/E**
- **Increased data availability**
- **Extensible to future data sharing and peripheral technology**

To implement the above objectives, Volume Management has changed keywords for some of the command interpreter commands. New operation commands have been added and some commands were deleted. Also MPE V/E's VINIT was replaced by a new utility called VOLUTIL.

3.0 Specifications

3.1 Media versus Devices

In HPE there is a distinction between the media and the device. For example, the device is the disc drive and the media is the disc pack. Volume management controls the disc media known as volumes.

Data structures pertaining to files, like the file label and extent information reside on the media. Access to files on the media is controlled through directories which also reside on the media. Volume set configuration, like the volumes within the volume set, the volume classes within the volume set, and the volumes within the volume classes, reside on the volume set media and NOT in the system configuration in the disguise of device classes.

Since volume sets are defined as a media concept and not a device concept, Volume Management views all volume sets the same whether the volume set is the system volume set or a non-system volume set. A media definition gives us a consistent view of volumes, volume classes, and volume sets. Another advantage of media definition is that the volume set definition moves with the volume set. This is unlike than MPE V/E which forces the volume set definition to be on each system using the volume set.

For MPE V/E user of FOPEN, device classes for discs will be transformed to volume classes. Also, LDEVs for discs will be transformed to a volume name. In the new intrinsic HPFOPEN, volume, volume class, or volume set may be specified for a file residing on a disc.

3.2 Volume Set Components

An example of the relationships in a four member volume set is shown in the following figure. The details are explained below.

Volume Set Relationships



Volume MEM1 is the master volume.

Volumes MEM1 & MEM2 are in volume class FOO.

Volumes MEM3 & MEM4 are in volume class FIE.

Volumes MEM2 & MEM3 are in volume class FOE.

3.2.1 Volumes

Volumes are currently one disc pack in HPE. They are members of a volume set. They may be members of volume classes. Each volume has a Free Space Map and a File Label Table. The Free Space Map controls the allocation of free space on the volume. The File Label Table contains file labels and file extent information. When a volume is created, it is given a volume ID which is a unique identifier.

Volume names are 16 characters long. The first character must be an alpha followed by alpha/numeric characters. The underbar, '_', is also allowed after the first character. All alpha characters are upshifted.

3.2.2 Volume Sets

A volume set in HPE can contain 1 to 255 members or volumes in its life time. Volume sets may contain 1 to 255 volume classes. A volume set must contain a master volume. The master volume contains the volume set configuration in the Volume Set Information Table, and the root directory in addition to the Free Space Map and the File Label Table. When the volume set is created, it is given a unique volume set ID.

Volume set names are 32 characters in length. The first character must be an alpha followed by alpha/numeric characters. The underbar, '_', and the period, '.', are also allowed after the first character. All alpha characters are upshifted. Some examples follow:

GONDMANALAND

ACCTS_PAYABLE

THE_3rd_example

PV.PUB.SYS

The HPE volume set name syntax supports the MPE V/E volume set name syntax. The MPE V/E volume set name is now a flat name and NOT a hierarchical directory name.

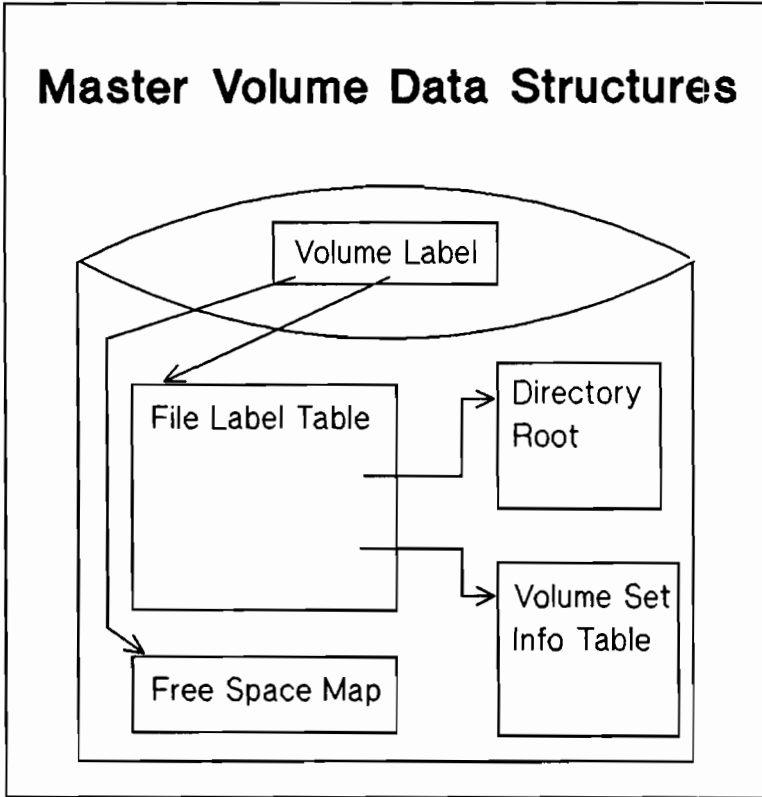
3.2.3 Volume Classes

Volume classes can contain 1 to 255 volumes. They are logical entities that control the allocation of disc storage to certain volumes. Volume classes do not need to exist within a volume set; but for compatibility purposes, the volume class DISC should be configured. Volumes may be in one volume class or in several.

Volume class names are 32 characters in length. The first character must be an alpha followed by alpha/numeric characters. The underbar, '_', and the period, '.', are also allowed after the first character. All alpha characters are upshifted. As with volume sets, the MPE V/E volume class syntax is supported in the HPE volume class syntax.

3.2.4 Master Volumes

The master volume is the only volume needed to define a volume set. As stated before, it contains the volume set configuration in the Volume Set Information Table for the volume set of which it is a member. It also contains the root directory for the volume set. See the figure below.



When the master volume is mounted, the volume set is considered mounted. Infact the master volume must be mounted before any file access can be made to other volume set members.

The master volume for the system volume set is special. It contains all the disc bootable images and system configuration. It must be mounted for the system to boot or run.

3.3 Data Availability

HPE Volume Management attempts to keep as much data as possible available to the system. This implies that if all the volumes in the volume set are not mounted, the user will be able to access the data from the available volumes in the volume set. HPE Volume Management easily allows the user to partition the data so that if a volume goes down, most of the files are still available.

Data partitioning is achieved by restricting where the files are built. There are three levels of data partitioning. They are:

- **Volume Set restriction**
- **Volume Class restriction**
- **Volume restriction**

The default volume restriction is the volume set of the group that the file is in. This means that the file extents are placed on any volume within the volume set. Note that a file cannot span a volume set. When a volume goes down, this restriction has the highest probability of stopping access to files. If the master volume goes down, then access to the entire volume set is denied for disc space allocation.

The next level of volume restriction is the volume class. The volume class restriction has to be specified at file creation time. Thus the file is placed only on the volumes within the volume class. If the volume class is a subset of the volume set, then the probability of a disc going down and preventing access to your data has been reduced. If the master volume goes down, then access to the entire volume set is denied for disc space allocation.

The most granular level of volume restriction is the volume. Again the volume restriction has to be specified at file creation time. The file extents are placed only on that volume. With this restriction, the probability of a disc going down and affecting access to data has been reduced further. The master volume must be up for space allocation.

Another type of restriction is to have non-system volume sets. It also allows the volume class and volume restrictions mentioned above. Multiple volume sets partition data very well. Non-system volume sets can easily be moved from HPE system to HPE system. Another advantage is that disc drives may be shared; for example, your development volume set is placed in the disc drives during the day, and your accounting volume set is placed in at night. Finally back-ups will be able to be done by volume sets. Having multiple small membered volume sets is like having multiple volume classes on one volume set, except that volume sets can be moved and backed-up separately.

4.0 User Interface

The user interface has been made consistent for both the system volume set and the non-system volume sets. Defaults for commands and intrinsics are the system volume set. Keywords have been added to commands to invoke a straight-forward meaning and to be consistent across commands. At this time, the MPE V/E keyword 'VS' has been deleted from all commands.

4.1 Directory Related Commands

The keyword 'ONVS' has been added to the following commands:

NEWACCT, ALTACCT, PURGEACCT

NEWGROUP, ALTGROUP, PURGEGROUP

REPORT; STORE

The 'ONVS' keyword specifies the name of the volume set where the action of the command is to occur. If the keyword is not specified, the system volume set, **HPE_SYSTEM_VOLUME_SET** is assumed.

The keyword 'HOMEVS' has been added to the following commands:

NEWGROUP, ALTGROUP

The 'HOMEVS' keyword specifies the name of the volume set where the files within that group are to be built. The system volume set is assumed if the keyword is not specified.

Here are some examples:

Building an account on a non-system volume set.

```
: NEWACCT doctor, who; CAP= ia,ba,sf,nd,gl,am,al
: NEWACCT doctor, who; ONVS= known_universe
```

The above example builds the account *doctor* on the system volume set and on the volume set *known_universe*. Note that the account must be built twice, once on the system volume set that you will access it from, and on the volume set where the account exists.

Building a group on a non-system volume set.

```
: NEWGROUP leela; CAP= ia,ba,sf,nd; &  
      HOMEVS= known_universe  
: NEWGROUP leela; ONVS= known_universe
```

The above example builds the group *leela* on the system volume set and on the volume set *known_universe*. Note that the group must be built twice, once on the system volume set that you will access it from, and on the volume set where the group exists.

Altering a group on a non-system volume set.

```
: ALTGROUP leela; ONVS= known_universe; FILES= 10000
```

The above example alters the file limit of the group on the non-system volume set. The file limit is only valid on the the non-system volume set. All other options like 'PASS=' are invalid when using the 'ONVS' keyword, because those options are currently set only in the system directory.

Storing off files from a non-system volume set.

```
: STORE @.@.@; *T; ONVS= known_universe
```

The above store command would store off all the files on the volume set *known_universe*. This makes volume set back-ups easy.

The following commands have been deleted:

NEWVSET, ALTVSET, PURGEVSET

LISTVS

This commands have been deleted because volume sets are not related to the directory. Their functions have been placed in the new utility VOLUTIL. NEWVSET has been replaced by VOLUTIL's NEWSET. ALTVSET is now ALTVOL in VOLUTIL. PURGEVSET is replaced by scratching the volume set master. LISTVS has been replaced by VOLUTIL's SHOWSET.

Currently in MPE V/E, a user is able to logon to the system even if the user's home group resides on the private volume. However in HPE, if the user's home group resides on a non-system volume, the user will not be able to logon until that volume set is mounted.

4.2 Operations Related Commands

The following MPE V/E commands are supported:

MOUNT, DISMOUNT, LMOUNT, LDISMOUNT

VSUSER, DSTAT, VMOUNT

The MOUNT, DISMOUNT, LMOUNT, and LDISMOUNT commands will support the MPE V/E volume set name syntax. However, volume class mounts will not be supported. The VSUSER, DSTAT, and VMOUNT commands will be supported exactly as they are on MPE V/E except that volume names are now 16 characters long instead of 8, and volume set names are 32 characters long instead of 27.

The following new commands have been added:

VSRESERVE, VSRELEASE, VSRESERVESYS, VSRELEASESYS

VSCLOSE, VSOPEN

The VSRESERVE command requests the console operator to put a volume set on line, if not spun up, and designates that volume set is in use. The VSRESERVE commands reserves the volume set between file opens for the user. VSRESERVE supports the HPE volume set name syntax, while MOUNT only supports the MPE V/E volume set name syntax.

The syntax is:

VSRESERVE [HPE volset name] [;GEN=num] [;WAIT=numsecs]

If no volume set is specified, then the request is for the home volume set of the user's logon group and account. Otherwise the user must specify the full volume set name. The generation number is the same as MPE V/E's. The WAIT parameter is used to specify the number of seconds to wait for the completion of the request before giving up and returning to the user. For example, it takes five minutes for a 7935 disc pack to spin up.

A VSRESERVE will not take place if the volume set is closed or is in a close pending state.

The VSRELEASE negates a previous VSRESERVE command. The syntax is:

```
VSRELEASE [HPE volset name]
```

If the volume set is not specified, then the home volume set of the user's logon group and account is used.

The VSRESERVESYS command reserves the volume set for the entire system. The syntax is:

```
VSRESERVESYS [HPE volset name] [;GEN=num]
```

The volume set name must be specified. There is no wait parameter since this is an operator command.

The VSRELEASESYS commands negates the previous VSRESERVESYS command. It does not negate a VSRESERVE command. The syntax is:

```
VSRELEASESYS [HPE volset name]
```

The volume set name must be specified.

The VSCLOSE command designates to the operating system that the volume set is going to be removed. This command replaces the MPE V/E DOWN command. This command will restrict access to the volume set. Any job/session that 1) has NOT done an explicit VSRESERVE/MOUNT on the volume set and 2) currently has NO files open on the volume set will be denied access to the volume set. The syntax is:

```
VSCLOSE [HPE volset name] [;NOW]
```

The volume set name must be specified. The VSCLOSE commands patiently waits until all the files have been closed on the volume set unless the 'NOW' keyword is specified. If the 'NOW' keyword is specified, all the users of the volume set will be aborted. A message will be printed out on the console when the volume set is closed and ready for removal.

The VSOPEN command opens a previously closed volume set. Its syntax is:

```
VSOPEN [HPE volset name]
```

The volume set name must be specified. After the VSOPEN command is issued, the volume set is ready for use.

4.3 VOLUTIL

The HPE Volume Utility Subsystem (VOLUTIL) provides maintenance and inquiry commands for managing system and non-system volume sets, volume classes, and volumes. Commands are provided for both individual volumes and volumes within volume sets and

volume classes. VOLUTIL enables volumes and volume sets to be initialized. Information about volumes, like free space, or creation date, can be displayed. Relationships of volume sets and volume classes can also be displayed.

The NEWSET command creates volume sets. The syntax is:

```
> NEWSET [SNAME=] sname [MASTER=] vname [LDEV=] ldev
      [PERM=] percentPerm [TRANS=] percentTrans
      [GEN=] genNumber
```

SNAME is the volume set name. MASTER is the volume name of the master volume. LDEV is the logical device where the volume is mounted. PERM is the percentage of total disc space of permanent disc space to be allowed of that disc. TRANS is the same except for transient space, or virtual space. GEN is the generation number. PERM, TRANS, and GEN can be defaulted. The defaults are zero.

The SHOWSET command displays information about the volume set. The command obtains information like creation date of the volume set, the volumes in the volume set, the volume class, the free map, and the HPE volume label.

To add a new class, VOLUTIL has the NEWCLASS command. The syntax is:

```
> NEWCLASS [CNAME=] [sname:] cname [MEMBERS=] vname [[, vname], ...]
```

CNAME is the volume class name, is can be made up from a volume set and a volume class name. VNAME is the name of the volumes that are a part of the volume class.

The SHOWCLASS command displays information about the volume class. The command obtains the same information as SHOWSET, except that it is class relative.

To add a volume to any volume set, try the NEWVOL command. The NEWVOL command syntax looks like this:

```
> NEWVOL [VNAME=] [sname:] vname [LDEV=] ldev
      [PERM=] percentPerm [TRANS=] percentTrans
      [[CLASSES=] cname [[, cname]. . .]]
```

The NEWVOL command creates a new volume. If the LDEV is not specified, an entry is placed in the Volume Set Information Table, and the volume is not initialized. Classes may be attached to the volume at this time.

The SHOWVOL command displays information about the volume in the same fashion as the SHOWSET command.

The above mentioned commands are not all the commands of VOLUTIL. There are commands to scratch volumes, unscratch them, alter classes, purge volumes, and condense

volumes. HPE commands may be executed from VOLUTIL. VOLUTIL has a logging feature, a redo facility like HPE's command interpreter, and has a script file facility.

Creating a non-system volume set with classes and volumes

```
> NEWSET SNAME=hpe_fs MASTER=gary LDEV=20
> NEWVOL VNAME=rick LDEV=21 PERM=100 TRANS=100
> NEWVOL VNAME=kendall LDEV=22
> NEWCLASS CNAME=vol_mgt VNAME=rick,kendall
```

The above example creates a volume set named `hpe_fs`. It contains three member volumes: `gary`, `rick`, and `kendall`. The volumes `rick` and `kendall` are in the volume class `vol_mgt`. Note that the volume `gary` is not in a volume class.

5.0 Summary

HPE Volume Management has made volume management consistent for both system and non-system volume sets. The user sees that the only difference between system and non-system volume sets is the fact that the system volume set is needed to run the system. The operator now has control over the right granularity, the volume set. Finally, HPE Volume Management is more robust than its predecessor. Not all the volumes in the volume set have to be mounted before the user is able to access files.

Acknowledgments

I would like to acknowledge Dean Coggins for coding volume recognition, Gary O'Neill for HPFOPEN code relating to volume management, Walt McCullough for wading through directory code, Dave Schoen for start-up code, Mark Diekhans for virtual space management code, Kendall Sutton for sorting through the new commands, and Howard Burrows for coding VOLUTIL. I would like to thank Al Kondoff and Sue Kimura for their support.

The System Manager's Toolbox

by: Blake, Isaac

We regret that this paper
was not received for
inclusion in these proceedings.

COMMUNICATING BETWEEN HPS AND FOREIGN SYSTEMS

Terry Atkinson,
M/A-COM Telecommunications, Inc.,
11717 Exploration Lane,
Germantown,
Md., 20874.

Introduction

For several years, our company had been using HP3000s for its business users, and computers from a different manufacturer for the engineering users. For the purposes of this paper, I shall refer to the two families of computers as the Montagu and Capulet families. The computers of one type were each linked together in a network, with mail communication between all the users, but with no communication between the two families. When we wanted to integrate our West Coast office into the system, it was no longer acceptable for the two families not to talk to each other. The East Coast officers were on one mail system, and the West Coast officers were on the other, with no means of communicating. Yet rapid communication facilities were essential. Telephone conversations were not an answer. With the time difference between the two coasts, and the hectic schedules of the officers, it turned out that the executives' secretaries spent more time talking to each other than did their managers. A computer connection was imperative.

A fully-automated, two-way link, connected to the mail systems of both computer systems, was set up. This paper gives detailed information on how the HP side of the link works. The information on the communication link between different families of computers is given separately from how this was linked to HPMAIL. Appendices give detailed descriptions of the file formats necessary to communicate with HPMAIL.

Initial Decisions

Before we started on the communication project, we made several basic decisions. Some of these were right and proved to be very useful. Others were not. I shall describe which decisions should have been made and why. Where our decisions could be improved on, I shall explain what factors would cause a different choice.

The two most fundamental decisions we made were that we would have only one computer of each type communicating with the other family, and that they would communicate in a standard format that would be easy to transmit and which was not specific to the mail systems of either end. The two computers we selected to communicate were Romeo from the Montagu family, and Juliet from the Capulet family. These were the only members of their families to talk to each other. The other computers

would transmit and receive via these two. Figure 1 shows this arrangement

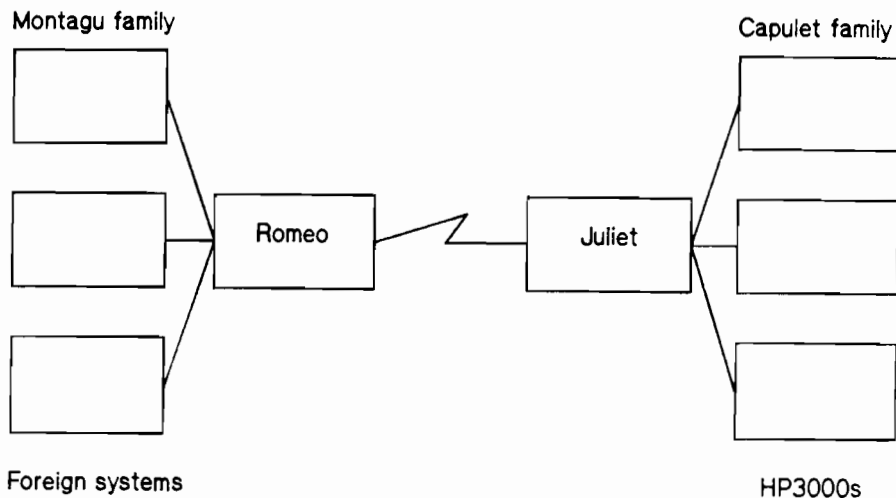


Figure 1

Initially we communicated via magnetic tape. Capulet computers would send messages to Juliet. There they would be transcribed into the standard format, and written to tape. The tape would be transferred to Romeo, the messages reformatted to Montagu standards, and then transmitted to the appropriate nodes. This worked, but with tremendous problems. The two families used different standards for just about everything, and certainly could not read tape labels from each other. This meant we could not magnetically protect our tapes, but relied on paper labels written by the operators. As any operator will realize, in the hectic atmosphere of a large DP shop, tapes got misplaced or were overwritten before they could be transferred. It was a daily routine to retransmit lost messages. An automated link was essential.

Neither of the computer families would talk directly to the other, but interestingly both were prepared to talk to an IBM computer via 2780/3780 protocol. In theory, if we had an IBM computer in the middle, both Romeo and Juliet would happily talk to it and allow it to pass on their messages. The question was, could they be persuaded to talk to each other without an IBM in the middle?

The answer was yes, and we set up a two-way fully-automated link using the 3780 RJE emulator of both computers. At this point we found out that we had made a wrong choice when we had defined the standard format. The idea of a standard format was a very wise one. The mail systems on both sides have been upgraded at least twice without affecting the link. In each case, the appropriate programmer has amended the interface, tested it on their side, and then the new version has been brought into production with few problems. However, when we defined the format we decided to make the records fixed length. We included three header records which defined the message (sender, recipient, subject, time and date stamp), and followed that with records containing 80 characters of text. CR LF markers occurred anywhere in the record. This was a mistake. When we switched to an automated link, we found that HP's DSN/RJE changed CR LF into an end-of-record marker, and split the record at that point. We had to write a subprogram on the HP to capture these broken records and put them back together again. Had the text originally been variable length, with a maximum length of 80 characters, ending with a CR LF, then it would have been much easier for everyone. As it is, the HP end of the link now has a special program to put the records back together, while the other end has a special program to break them up!

Preliminaries

Before getting into a detailed discussion of the communication link, there are a few things to explain first. RJE and HPDESK were never intended to operate together. Each system has its own account, and certain functions must work in particular groups of those accounts. The RJE transmission takes place in MSG.RJE, while the mail transport uses HPMAIL.HPOFFICE and MAILDB.HPOFFICE. For the link to work, each software system must be able to read files created by the other. Therefore the two accounts must have ACCESS set to allow "READ=ANY", and these groups must have the same ACCESS setting. For security reasons, it is worth ensuring that other groups in these accounts have read access restricted to the account only.

The communication link described in this paper uses only the mail functions of HPDESK. For consistency and clarity, when I am referring generally to the mail systems I shall use the term HPMAIL. Where I refer to a specific version of a package, I shall use whatever name is given to the particular version.

The mail link uses a large number of files to operate. There are many different processes necessary for the system to run, and it is necessary to pass control information between them. Therefore the link makes very heavy use of IPC files. The distinguishing features of these files are: a file can be read by one process at the same time as it is written to by another; records are read from the top of the file and written to the end; when a record is read, it is automatically deleted; it is possible for a program to issue a read instruction, and then hang



until there is a record written to the file for the program to read; and it is possible to get around this hanging read by using the FFILEINFO intrinsic, with the EOF option, so as to find out whether or not there are any records in the file before issuing a read.

When the HP mail system creates messages for transmission to other systems, it uses a special naming convention for the files containing messages. The format is Edddhhmm, where ddd is the day of the year when the file was created, hh is the hour, and mm the minute of creation. These are referred to as E-files. This convention is heavily used by the link system so as to ensure there are no duplicate file names. This system uses C-files, R-files, and V-files, all with the same format as the E-files. There are also M-files with a similar format, but with the difference that the second of creation had to be included. The format of the name of an M-file is Mdhmmss.

RJE

The automated transfer uses the standard IBM-compatible 3780 RJE package, transmitting in EBCDIC. (The reason for using EBCDIC is that the RJE for the Montagu family did not support ASCII). Interface programs are used to arrange messages for processing by the RJE system, and particular options have been selected with the packages, but nothing non-standard has been done to the packages.

A sample RJE job stream is shown in Figure 2. Since the HP RJE package believes it is being controlled by an IBM computer, it uses the command RJOUT to read (take information OUT of the IBM), and the command RJIN to send (to take IN to the IBM).

```

|JOB RJELINK,MGR/.RJE/;HIPRI;OUTCLASS=LP
|FILE RJLFILE=Rdddhhmm.MSG.RJE;DEV=DISC;DISC=64000,32,1
|BUILD *RJLFILE;REC=-82,,F,ASCII
|FILE RJLIPCIN=RJLIPCIN.MSG.RJE
|CONTINUE
|RJE
#RJLINE 3780;LINECODE=EBCDIC;MSGFILE=RJLIPCIN
#RJOUT @RJLTRAN(P);WAIT=480;REPEAT=YES;INTERRUPT=YES
#RJEND
|RUN RJLTOHP.PROG
|EOJ

```

Figure 2

The link is two way, yet uses one wire. Like the Eagles in "Hotel California", the RJE links can say "We are programmed to receive". Each end is hanging on a read from the other end. On the HP side it is necessary to interrupt the system in order to send a message. The RJOUT command has to have the parameter "INTERRUPT=YES" set. This means that the RJLINE command allows an IPC file to be defined (which

must be in MSG.RJE) which RJE will read. This file is to contain the names of files which contain commands which RJE is to process. Unfortunately this parameter only works if the parameter "REPEAT=YES" is also used. This allows automatic continuation of the link, but it makes it impossible to do anything else with the link. Since the RJOUT receive command is automatically repeated, it is not possible to process any messages that are received. It is also impossible to terminate the link except by aborting it, which is not desirable since it might be in the middle of transmitting or receiving a message.

The solution to this is to write a procedure program (RJLTRAN in this example) and to define it on the RJOUT line. This procedure does any necessary processing on the received message and writes it to an R-file, and then initiates a command to terminate the RJE system. An important feature of our link is that there is a file in MSG.RJE which has one record in it. This record is an RJOUT command for RJE, which has the parameters WAIT=0,1;REPEAT=NO;INTERRUPT=NO. This commands RJE not to be interrupted, not to repeat, and to wait only one second before terminating. RJLTRAN writes the name of this file to the RJLIPCIN IPC file defined on the RJLINE command, and causes RJE to terminate. The problem with this method is that the timeout is considered an error. For this reason, we have to put a CONTINUE statement before the RJE command. Otherwise the timeout causes the rest of the job stream to be ignored. The timeout also causes an error report to be printed on the RJE standard list. Operators get so used to seeing an error, and ignoring it, that this tends to mask cases which are genuinely errors.

This method is also programatically repeated in the MAILOFF routine, so as to bring down RJE when HPMAIL is brought down. Since we have this method of bringing down RJE when necessary, we are able to put an 8 hour timeout (almost the maximum value) on the normal RJOUT command.

When RJE has terminated, the job stream initiates a program which checks the status of the link. It checks the EOF on the R-file to see if any records have been written to it. If so, then it knows a message has been received and it streams the job to process the message. If not, then it knows no message has been received, and it purges the R-file. Then the job reads in the RJE job stream, changes the name of the R-file in the FILE statement so as to avoid duplicate file names, and rewrites the job stream. It then checks to see if RJE is being brought down. Another IPC file (RJLIPCON) is used as a switch for this. When HPMAIL and RJE are first started, a record is written to the RJLIPCON switch file. When RJE is to be brought down, the record is read from the IPC file, thus deleting it. RJLTOHP, in the RJE job stream, checks EOF on RJLIPCON to see if there are records in it. If so, then RJE is to stay up, so the program restreams the RJE job stream. If there are no records in RJLIPCON, then the program allows the RJE job stream to terminate. Figure 3 shows this arrangement.

Connecting to Mail

The detailed description of how RJE and Mail are connected is given in two parts - first sending from HPMAIL across RJE to Romeo, second receiving from RJE and importing the messages into HPMAIL. Figure 4 gives a simplified picture of how the link works in total.

Mail FSC

The communication system uses the FSC ARPA format to export mail to the other computers. This is one option of HP's standard format for sending messages to non-HP computers. FSC stands for Foreign Service Connection. Although HP uses a published standard for this format, it is insufficiently detailed to write decoding programs. Details of the FSC ARPA format are given in Appendix A. This Appendix also contains a photocopy of an HPMAIL message converted to ARPA format.

HPMAIL creates E-files (in HPMAIL.HPOFFICE) containing the mail messages, and writes the names of the files to an IPC file defined in the EFT/FSC screen of MAILCONFIG (our file is FSCIPC.HPMAIL). Once an hour, our system initiates an interface program which checks EOF on the IPC file. If there are no records, the program ends. If there are records, then the program processes all the E-files.

The program reads the E-files in sequence and converts the mail messages to the standard format. The messages are written to a V-file, which has the same date and time stamp in its name as the first E-file processed. This was accidental, and done solely to guarantee a unique file name. However, it has proved very useful in tracing messages when this has been necessary. The program then creates an RJIN command containing the name of this V-file, and writes this record to a C-file with the same date and time stamp. The name of this C-file is then written to the RJIIPCIN message file defined to RJE to be used for interrupting the system. For this reason, the program must run in MSG.RJE, although it reads from HPMAIL.HPOFFICE. Figure 5 illustrates this.

Mail EFT

The communication link was first written before FSC was available, and EFT format was used in both directions. When FSC format came out, it was our intention to use that in both directions. However, FSC does not support the REPLY option in HPMAIL. Our users had got used to REPLYing to messages from the Montagu computers, and we could not take this away. So we had to continue to use EFT format for importing messages.

EFT (External File Transfer) format is intended to be used for communications only between two HP computers, both of which have HPMAIL.

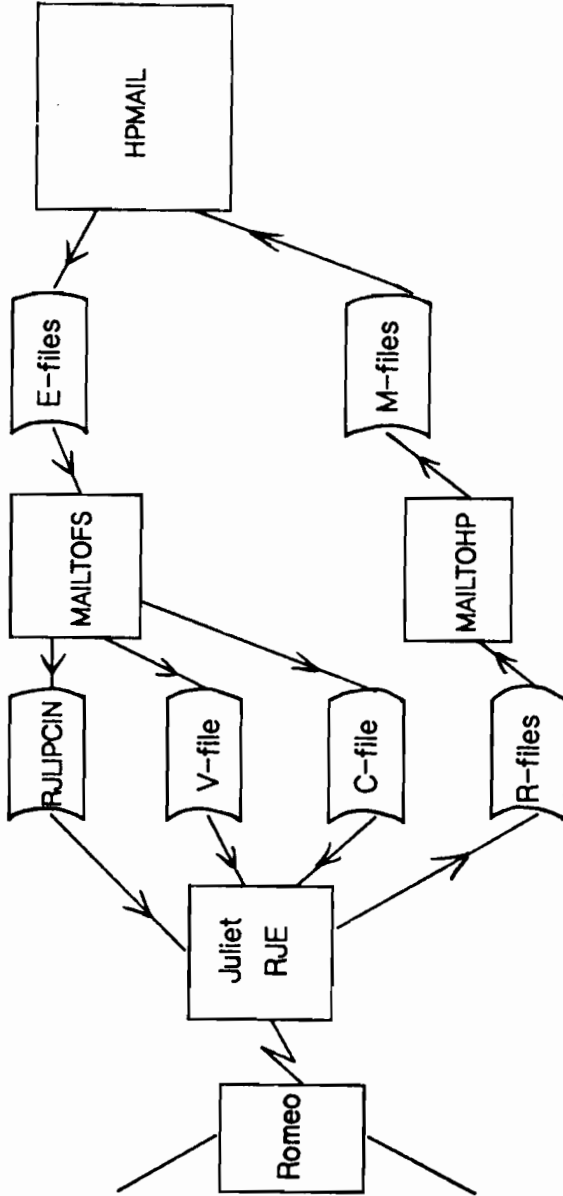


Figure 4

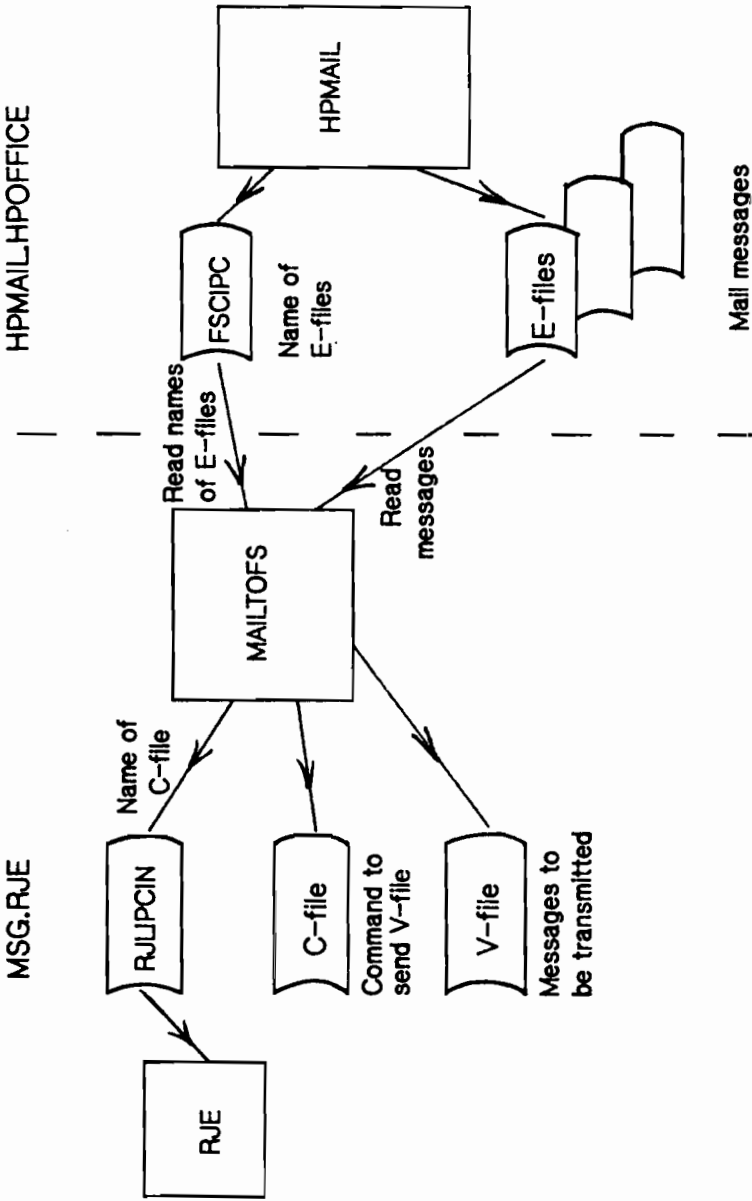


Figure 5

For this reason, HP has not published descriptions of this format (a useable description of EFT format is given in Appendix B, together with the EFT version of the same message used as an example for the FSC format), and do not guarantee to keep it constant. However, since it must be used to communicate between HP computers running different versions of HPMAIL, it is safe to assume that it will not change. We have already gone from HPDESK 1 to HPDESK 3.1 with no problems.

When the RJE job stream finds there is a received message to send to the HP, a mail job stream is streamed in the HPOFFICE account. Before starting the job, the RJLTOHP program changes the mail job stream so that it contains the name of the R-file to process. Thus the job stream has to be in the RJE account, although it runs in the HPOFFICE account. Figure 6 contains an example of the job stream.

```
|JOB MAIL2HP,MGR/.HPOFFICE/,MAILDB;HIPRI;OUTCLASS=LP;PRI=CS
|FILE EFTMAIL=Rdddhmm.MSG.RJE
|FILE MAILWORK=MAILWORK;DISC=6400,32,1
|FILE MAILFILE=Mdhmmss;DISC=32000,32,1
|RUN MAIL2HP.MAILPROG
|RUN DSSEFT.HPMAIL.SYS;LIB=G
|EOJ
```

Figure 6

The MAIL2HP program converts messages to EFT format. This is a very complex format. In several places there are counts of the records that follow. So it is necessary to write records to a work file while counting them. Once all records of that type have been processed, the count can be placed in the earlier record, which is then written out to disc. The subsequent records are then read from the work file, and written out in their turn.

The Mail messages are written to an M-file. Since messages can be received in quick succession, it was necessary to include seconds in the name of the file so as to avoid duplicate file names. The name of the M-file is then written to an IPC file (EFTIPCIN.MAILDB.HPOFFICE), which is defined by HPMAIL. The job stream then runs DSSEFT to import the messages into HPMAIL. Figure 7 illustrates this.

Advantages and Disadvantages

The communication system is very stable. Hundreds of messages are sent in each direction every week, with very few problems. Some of the problems that do exist are procedural, although one was a problem with the HPMAIL software. The early version of the FSC option would occasionally fail to include recipients' names in a message. It tends to be sort of difficult to deliver mail if you do not know to whom it is supposed to go. This was corrected in HPDESK version A.03.01, which is

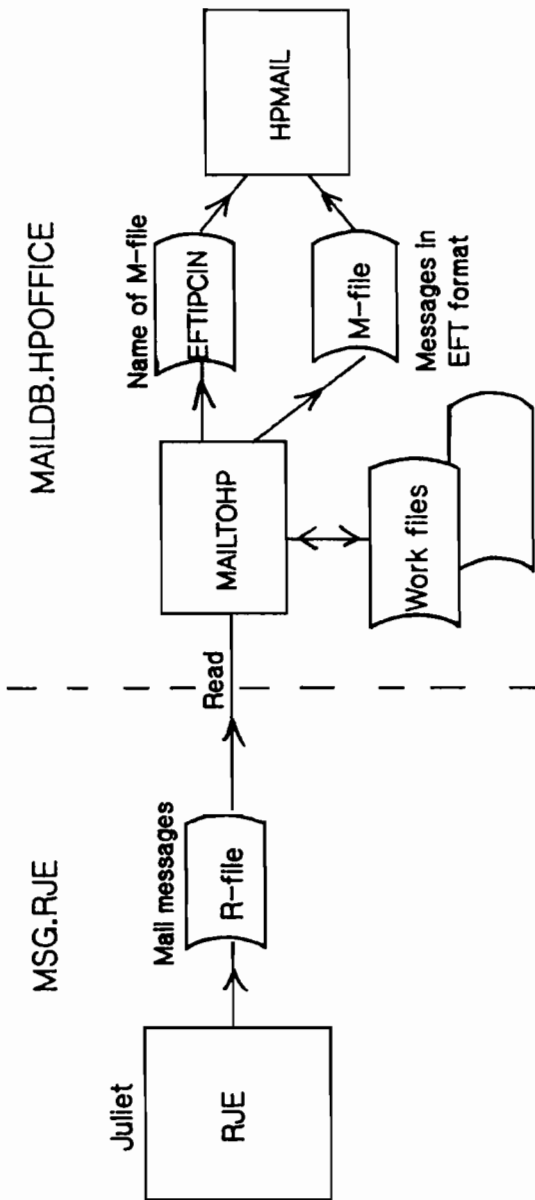


Figure 7

available with T-delta-4 and some delta version of UMIT. It would be worthwhile installing this version of HPDESK before attempting to export messages in FSC ARPA format.

Some procedural items to be aware of are:

- It is not possible to designate a message as "private" if it is being sent to an FSC recipient. This can cause the message to be lost until the next MAILMAINT run, when it will suddenly be flagged as an error.
- Be careful if users start sending binary files across the link. The special control characters in Wordstar succeeded in crashing the HP RJE system.
- If the HP receives a very large volume of messages all at once, it will cause the programs to run literally for hours, and for HPMAIL to become so slow that users complain. If the processing of the messages can just be allowed to run its course, the system will return to normal.

The communications link has become an essential part of the way our company does business. It has proved itself so important to the company that when we moved the computers to a new computer room, bringing up the mail link was the top priority. The East Coast managers read their mail when they arrive at work in the morning, and send any messages to the West Coast. Three hours later, the West Coast managers arrive, read their mail, and send replies to the East. Before they go home, the East Coast managers read their mail and reply to messages from the West. Three hours later, the West Coast people do the same back. Their messages are ready and waiting when the East Coast starts work again the next day.

APPENDIX ARules for ARPANET Format Files

Some explanation of ARPA format files is given in the HP manual, "Programmatic Access to HPDESKMANAGER" (Part No. 36570-90040). This information is insufficient for writing programs to process the files. Also, the example of an ARPA file given in the manual (on page 5-12) is of ARPA Compressed format. The rules below are for basic ARPA format, and are intended to be read in conjunction with the HP manual. Information that is in the manual is not repeated here.

1. Keywords in the header are:

Date
Sender
To
CC
BCC
From
Subject
X-HPDESK-ID
X-HPDESK-PRIORITY

These are the only keywords that appear in the ARPA file header, and these keywords will always be EXACTLY as listed above, including upper and lower case as written above. The keywords, if present, will also appear in the order given.

2. All keywords in the header are preceded by CRLF.

3. Keywords in the header are followed by an indefinite number of blank spaces. After this, there is always a colon, one space, and then the data.

4. The date is in the format dd Mmm yy, where dd and yy are numeric, and Mmm is alphabetic (e.g., 16 Jan 86). If the date is earlier than the tenth of the month, the leading zero is replaced with a blank (e.g., 5 Jun 86).

5. Mailbox names and nodes in the Sender, To, CC, BCC, and From modules of the header are in a standard format. For the actual name, a firstname and middle name are optional, and the length of the name varies. For the mailbox node, sublocation is optional. With these provisos, the names will always be in the format:

"Firstname Middlename LASTNAME"@[Location/sublocation]

where punctuation, case, etc., will always be EXACTLY as shown.

While the manual "Programmatic Access to HPDESKMANAGER" states that sublocations are optional, it is in fact impossible to define a remote connection that does not have a sublocation. Therefore sublocation is, in reality, mandatory on mail going out from the HP.

6. One E-file will contain mail messages for only one foreign node. If one message is to go to users on several different nodes, one E-file will be created for each node.

7. Some header modules, as defined by the keywords, are optional. All messages will contain Date, Sender, Subject, and the X-HPDESK- keywords, although there might not actually be a subject (sender replies with a carriage return to the prompt for Subject). The recipient keywords and modules are optional, and will be present only if there is a recipient in that class on the receiving machine. For example, if the sender enters the name of a remote user in reply to the prompt for From, and if that user is the only recipient on that remote system, then a record will be created with no recipients, but with a From module included.

An example of a file in FSC format follows. To allow the largest possible type face for the listing, the first page shows the octal representation of the file, and the next page gives the character interpretation. The file was created on node DCC to be sent to node LUKE.


```

E1561615.HPMAIL RECORD 0 (%0, #0)
00000: 106015 005104 060564 062440 020040 020040 035040 020085 020112 072556 020070
00014: 033040 030466 035060 031440 043515 052015 051233 062556 062145 071040 020040 020040
00030: 035040 021124 062562 071171 020101 052113 044516 051517 047042 040133 042103 041457
00044: 030061 056415 005124 067440

E1561615.HPMAIL RECORD 1 (%1, #1)
00000: 020040 020040 020040 020040 035040 021124 062562 071171 020101 052113 044516 051517
00014: 047042 040133 046125 045505 027460 030535 068412 052565 061152 062543 072040 020040
00030: 020072 020105 074141 066560 066145 020157 063040 043123 041440 060556 062040 042506
00044: 052040 063157 071155 060564

E1561615.HPMAIL RECORD 2 (%2, #2)
00000: 071440 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
00014: 020040 020040 020040 020040 005130 026510 050104 042523 045455 044504 035040 030460
00030: 032064 030062 030467 020060 020060 020060 020042 046125 045505 020040 030061 021015
00044: 005130 026510 050104 042523

E1561615.HPMAIL RECORD 3 (%3, #3)
00000: 045455 050122 044517 051111 052131 035040 031415 005015 005040 006412 046545 071563
00014: 060547 062456 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
00030: 051545 067144 062562 035040 020040 020040 020104 060564 062544 035040 030066 027460
00044: 032457 034066 020141 072040

E1561615.HPMAIL RECORD 4 (%4, #4)
00000: 030466 030063 027015 005123 072542 065145 061564 035040 042570 060555 070154 062440
00014: 067546 020106 051503 020141 067144 020105 043124 020146 067562 065541 072162 066412
00030: 051545 067144 062562 035040 020124 062562 071171 020101 052113 044516 051517 047040
00044: 027440 042103 041457 030061

E1561615.HPMAIL RECORD 5 (%5, #5)
00000: 020040 020040 020040 020040 020040 020040 020040 020040 041557 067164 062556 072163
00014: 035040 031056 066412 020015 005120 060562 072040 030456 066412 020015 005040 020124
00030: 047472 020124 062562 071171 020101 052113 044516 051517 047040 027440 046125 045505
00044: 027460 030415 005040 066412

E1561615.HPMAIL RECORD 6 (%6, #6)
00000: 050141 071164 020062 027015 005040 006412 052150 064563 020151 071440 060556 020145
00014: 074141 066560 066145 020164 067440 071550 067567 020167 064141 072040 060556 020110
00030: 050115 040511 046040 066545 071563 060547 062440 066157 067553 071440 066151 065545
00044: 020151 067040 043123 041440

E1561615.HPMAIL RECORD 7 (%7, #7)
00000: 040522 030101 066412 063157 071155 060556 026040 060556 062040 064556 020105 043124
00014: 020111 067164 062562 067141 066040 043157 071155 060564 027015 005040 006412 040440
00030: 061154 060556 065440 066151 067145 020150 060563 020152 072563 072040 061145 062556
00044: 020151 067163 062562 072145

E1561615.HPMAIL RECORD 8 (%10, #8)
00000: 062034 020141 067144 020164 064151 071440 064563 020164 064145 020145 067144 020157
00014: 063040 072150 062440 066545 071563 060547 062456 020040 020040 020040 020040 020040
00030: SAME: TO 000050-1
    
```

E1561615.HPMAIL RECORD 0 (%0, #0)
00000: Date: 5 Jun 86 16:03 GMT..Sender : "Terry ATKINSON"@[DCC/01]..To
E1561615.HPMAIL RECORD 1 (%1, #1)
00000: : "Terry ATKINSON"@[LUKE/01]..Subject : Example of FSC and EFT format
E1561615.HPMAIL RECORD 2 (%2, #2)
00000: s ..X-HPDESK-ID: 10440217 0 0 0 "LUKE 01" ..X-HPDES
E1561615.HPMAIL RECORD 3 (%3, #3)
00000: K-PRIORITY: 3.....Message.
E1561615.HPMAIL RECORD 4 (%4, #4)
00000: 1603...Subject: Example of FSC and EFT formats..Sender: Terry ATKINSON / DCC/01
E1561615.HPMAIL RECORD 5 (%5, #5)
00000: Contents: 2....Part 1.... TO: Terry ATKINSON / LUKE/01...
E1561615.HPMAIL RECORD 6 (%6, #6)
00000: Part 2....This is an example to show what an HPMAIL message looks like in FSC
E1561615.HPMAIL RECORD 7 (%7, #7)
00000: ARPA..format, and in EFT Internal Format....A blank line has just been inserte
E1561615.HPMAIL RECORD 8 (%10, #8)
00000: d, and this is the end of the message.

APPENDIX BHPMAIL EFT Internal Format

HPMAIL EFT Internal Format uses 10 different record types. These are identified as record types 0 - 9. Record types 0 - 2 are file headers that appear once at the beginning of an internal format file. Record types 3 - 8 are headers for each individual mail message. Record types 3, 4, and 7 are similar, although not identical. Record type 9 contains the text of the message. The records are 256 character fixed length records.

Record types 0 - 8 contain codes. The meanings of many of these codes have been identified and are described in this appendix. However, many other meanings have not been identified. It is quite possible that, for some of these, they have no meaning but contain whatever was in the buffer when the records were created. In some cases the transport system will not work if the codes are changed in any way, while other codes appear to have no effect on the operation of the HPMail system.

The HPMail heading records contain many counts of items that follow the header records. Therefore, when creating a file in EFT internal format, it is necessary to store information on work files until all the counts have been made. Then the header records can be created, and following detail records written after the headers

As far as can be ascertained, the purposes of the 10 record types are:

- 0 Information to identify the communicating computers
- 1 Starting position of mail messages
- 2 Continuation of record type 1
- 3 Sender information
- 4 Sender information
- 5 Recipient header information
- 6 Recipient list
- 7 Sender information
- 8 Text header information
- 9 Text

Record Formats

The formats for the record types are given with byte counts being used for offsets, beginning with 0 for a record. The terms "word" or "Dword" (doubleword) are used to identify items of 2 or 4 bytes in length respectively, and beginning on a word or doubleword boundary. The symbol % means the number is an octal number. The symbol b means a blank space is required. The formats below give details for communicating from a foreign computer called ROMEO, to accounts in location JULIET/01.

Record 0

| Starting Position | Length | Value | Meaning |
|-------------------|--------|----------|-------------------------------------|
| 0 | 1 word | %1 | file type - required |
| 2 | 6 | JULIET | Location of receiving mail node |
| 8 | 2 | 01 | Sub-location of receiving mail node |
| 10 | 8 | ROMEObbb | Name of sending computer (node) |

The rest of the record is blank filled.

Records 1 and 2 combine to provide 128 doublewords (64 on each record) that are used to record the starting record number of each mail message on the file. The starting record number of the first record is always 3, and this value is stored in the doubleword beginning at position 0 in record 1. The last doubleword on record 2 is always 0. This leaves space for 127 message identifiers. This is the reason that a file in HPMail EFT internal format cannot contain more than 127 messages.

Record 3

The first 4 words of record type 3 contain codes whose meanings are unknown. Words 1, 3, and 4 are always the same on all occurrences of record type 3. Word 2 starts off with different values on files created on different occasions, but increases by %23 with each occurrence of record type 3, or its related record types 4 and 7. The doubleword from 68 - 71 always contains the same value on one file, but this value changes for different files.

| Starting Position | Length | Value | Meaning |
|-------------------|----------|---|----------------------------------|
| 0 | 1 word | 0 | |
| 2 | 1 word | Varies, increases by %23 for each occurrence of record type 3, 4, or 7. | Meaning not known. |
| 4 | 1 word | %177634 | not known - required |
| 6 | 1 word | %1 | not known |
| 8 | 60 | | Title of message |
| 68 | Dword | varies | not known |
| 72 | Dword | %0 | |
| 76 | Dword | | Number of seconds since 1/1/61 |
| 80 | 2 Dwords | %0 | |
| 88 | 36 | | Sender's account name |
| 124 | 6 | | Name of sender's location (node) |
| 130 | 2 | | Sender's sub-location |

| | | | |
|-----|---------|-------|-----------|
| 132 | 7 words | %0 | |
| 146 | 1 word | %2 | not known |
| 148 | Dword | %0 | |
| 152 | 2 words | %2,%2 | not known |

The rest of the record is filled with %0.

Record 4

| Starting Position | Length | Value | Meaning |
|-------------------|---------|-----------------------|--------------------------------|
| 0 | 1 word | %0 | |
| 2 | 1 word | %23 | more than record 3 - not known |
| 4 | 1 word | %2216 | not known |
| 6 | 1 word | %0 | |
| 8 | 12 | DISTRIBUTION | |
| 20 | 48 | Blank spaces | |
| 68 | 78 | Same as record type 3 | |
| 146 | 1 word | %0 | |
| 148 | 3 words | Same as record type 3 | |
| 154 | 1 word | %0 | |

The rest of the record is filled with %0.

Record 5

| Starting Position | Length | Value | Meaning |
|-------------------|----------|-----------------|---|
| 0 | 14 | \$OLDPASSOFFICE | |
| 14 | 1 word | %421 | not known |
| 16 | 1 word | %60 | not known |
| 18 | 1 word | %0 | |
| 20 | 1 word | %6 | not known |
| 22 | 2 words | %0 | |
| 26 | 1 word | %1 | not known |
| 28 | 1 word | %34 | not known |
| 30 | 1 word | %400 | not known |
| 32 | 1 word | %177720 | not known |
| 34 | 1 word | %2216 | not known |
| 36 | 2 Dwords | | Count of number of recipients of message. (Count is stored twice, in consecutive Dwords.) |
| 44 | 1 word | %176400 | not known |
| 46 | 1 word | %1 | not known |
| 48 | 1 word | %0 | |
| 50 | 1 word | %20 | not known |
| 52 | 3 words | %0 | |
| 58 | 1 word | %6 | not known |

The rest of the record is filled with %0.

Record 6 contains the names and locations of the recipients of the message. The information is stored in segments that are 48 characters long. If there are more recipients than will fit on one record, the segments are continued onto another record without regard for record boundaries. For example, if there are 7 recipients, then the first 5 segments, and the first 16 characters of the 6th segment, are on the first record. The last 32 characters of the 6th segment are in positions 0 - 31 of the second record. The 7th segment begins in position 32 of the second record. 11 recipients require three records, and so on.

The format of the segments is given below:

| Starting Position | Length | Value | Meaning |
|-------------------|--------|----------------------|--|
| 0 | 1 word | %300 %100 | 1st segment 2nd or subsequent segment |
| 2 | 1 word | %1 %2 %3 | Primary recipient CC recipient BCC recipient |
| 4 | 36 | LASTNAME,OTHER NAMES | recipient's name |
| 40 | 6 | | Recipient's location |
| 46 | 2 | | Sub-location |

Record 7

| Starting Position | Length | Value | Meaning |
|-------------------|--------|-------|---|
| 0 | 1 word | %0 | |
| 2 | 1 word | %23 | more than record 4 - not known |
| 4 | 1 word | %2217 | not known |
| 6 | 1 word | %0 | |
| 8 | 60 | | Title of message |
| 68 | 84 | | Same as record type 4 |
| 152 | 1 word | | Number of record types 8 and 9 in this message |

The rest of the record is filled with %0.

Record 8

| Starting Position | Length | Value | Meaning |
|-------------------|--------|-----------|-----------|
| 0 | 8 | \$OLDPASS | |
| 8 | 20 | %0 | |
| 28 | 1 word | %34 | not known |
| 30 | 1 word | %400 | not known |

| | | | |
|----|---------|--------------------------------|-----------|
| 32 | 1 word | %177660 | not known |
| 34 | 1 word | %2217 | not known |
| 36 | 1 word | %0 | |
| 38 | 1 word | Number of 80-character message | lines in |
| 40 | 1 word | %0 | |
| 42 | 1 word | %1777 | not known |
| 44 | 1 word | %175400 | not known |
| 46 | 1 word | %10 | not known |
| 48 | 1 word | %0 | |
| 50 | 1 word | %20 | not known |
| 52 | 3 words | %0 | |
| 58 | 1 word | %12 | not known |

The rest of the record is filled with %0.

Record 9 contains the text of the message. It is stored in the records as continuous strings of 80 characters in length. There are no control characters such as carriage returns or line feeds. Short lines are padded to 80 characters with blank spaces. When the end of a record is reached, the 80 character line runs over onto the next record without regard for record boundaries. Once all text has been included, the last 256 character record is filled with blank spaces.

An example follows of a file in EFT format. This is the same message that was used in Appendix A as an example of the FSC format. Again, to allow the largest possible type face for the example, the octal representation is given first, and then the character interpretation.

```

M7084804.MAILDB.RECORD.0 (%0, #0)
00000: 000001_042103 041440 020040 030061 046125 045505 020040 020040 020040 020040 020040 020040
00014: SAME: TO 000200-1

M7084804.MAILDB.RECORD.1 (%1, #1)
00000: 000000_000003 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000
00014: SAME: TO 000200-1

M7084804.MAILDB.RECORD.2 (%2, #2)
00000: 000000_000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000
00014: SAME: TO 000200-1

M7084804.MAILDB.RECORD.3 (%3, #3)
00000: 000000_002350 177634 000001 020040 020105 074141 066560 066145 020157 063040 043123
00014: 041440 060556 062040 042506 052040 063157 071155 060564 071440 020040 020040 020040
00030: 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 000000 000001
00044: 000000 000000 027723 034210 000000 000000 000000 040524 045511 047123 047516
00060: 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
00074: 020040 020040 046125 045505 020040 030061 000000 000000 000000 000000 000000 000000
00110: 000000_000002 000000 000000 000002 000002 000000 000000 000000 000000 000000 000000 000000
00124: SAME: TO 000200-1

M7084804.MAILDB.RECORD.4 (%4, #4)
00000: 000000_002373 002216 000000 042111 051524 051111 041125 052111 047516 020040 020040
00014: SAME: TO 000030-1
00030: 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 000000 000001
00044: 000000 000000 027723 034210 000000 000000 000000 040524 045511 047123 047516
00060: 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
00074: 020040 020040 046125 045505 020040 030061 000000 000000 000000 000000 000000 000000
00110: 000000_000000 000000 000000 000002 000000 000000 000000 000000 000000 000000 000000 000000
00124: SAME: TO 000200-1

M7084804.MAILDB.RECORD.5 (%5, #5)
00000: 022117 046104 050101 051523 047506 043111 041505 000421 000060 000000 000006 000000
00014: 000000 000001 000034 000400 177720 002216 000000 000001 000000 000001 176400 000001
00030: 000000 000020 000000 000000 000000 000006 000000 000000 000000 000000 000000 000000
00044: SAME: TO 000200-1

M7084804.MAILDB.RECORD.6 (%6, #6)
00000: 000300 000001 040524 045511 047123 047516 026124 042522 051131 020040 020040
00014: 020040 020040 020040 020040 020040 020040 020040 020040 042103 041440 020040 030061
00030: 020040_020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
00044: SAME: TO 000200-1

M7084804.MAILDB.RECORD.7 (%7, #7)
00000: 000000_002416 002217 000000 020040 020105 074141 066560 066145 020157 063040 043123
00014: 041440 060556 062040 042506 052040 063157 071155 060564 071440 020040 020040 020040
00030: 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 000000 000001
00044: 000000 000000 027723 034210 000000 000000 000000 040524 045511 047123 047516
00060: 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
00074: 020040 020040 046125 045505 020040 030061 000000 000000 000000 000000 000000 000000
00110: 000000_000000 000000 000000 000007 000000 000000 000000 000000 000000 000000 000000 000000
00124: SAME: TO 000200-1

```


M7084804.MAILDB RECORD 8 (%10, #8)
 00000: 022117 046104 050101 051523 000000 000000 000000 000000 000000 000000 000000 000000 000000 000000
 00014: 000000 000000 000034 000400 177660 002217 000000 000023 000000 001777 175400 000010
 00030: 000000 000020 000000 000000 000000 000012 000000 000000 000000 000000 000000 000000 000000 000000
 00044: SAME: TO 000200--1

M7084804.MAILDB RECORD 9 (%11, #9)
 00000: SAME: TO 000044--1
 00044: 020040 020040 020040 020040 054055 044120 042105 051513 026511 042072 020061 030064
 00060: 032060 031061 033440 030040 030040 030040 021114 052513 042440 020060 030442 020040
 00074: SAME: TO 000110--1
 00110: 020040 020040 020040 020040 020040 020040 020040 020040 054055 044120 042105 051513
 00124: 028520 051111 047522 044524 054472 020063 020040 020040 020040 020040 020040 020040 020040 020040
 00140: SAME: TO 000200--1

M7084804.MAILDB RECORD 10 (%12, #A)
 00000: SAME: TO 000110--1
 00110: 046545 071563 060547 062456 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
 00124: 020040 020040 020040 020040 020040 020040 020104 060564 062544 035040
 00140: 030066 027460 032457 034066 020141 072040 030466 030063 027040 020040 020040 020040 020040 020040
 00154: 020040 020040 020040 020040 051565 061152 062543 072072 020105 074141 066560 066145
 00170: 020157 063040 043123 041440 060556 062040 042506 052040

M7084804.MAILDB RECORD 11 (%13, #B)
 00000: 063157 071155 060564 071440 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
 00014: SAME: TO 000030--1
 00030: 051545 067144 062562 035040 020124 062562 071171 020101 052113 044516 051517 047040
 00044: 027440 042103 041457 030061 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
 00060: 041557 067164 062556 072163 035040 031056 020040 020040 020040 020040 020040 020040 020040 020040
 00074: SAME: TO 000140--1
 00140: 020040 020040 020040 020040 020040 020040 020040 020040 050141 071164 020061 027040
 00154: 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
 00170: SAME: TO 000200--1

M7084804.MAILDB RECORD 12 (%14, #C)
 00000: SAME: TO 000060--1
 00060: 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 052117 035040 052145
 00074: 071162 074440 040524 045511 047123 047516 020057 020114 052513 042457 030061 020040
 00110: SAME: TO 000200--1

M7084804.MAILDB RECORD 13 (%15, #D)
 00000: 020040 020040 020040 020040 020040 020040 020040 020040 050141 071164 020062 027040
 00014: 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040
 00030: SAME: TO 000124--1
 00124: 020040 020040 020040 020040 052150 064563 020151 071440 060556 020145 074141 066560
 00140: 066145 020164 067440 071550 067567 020167 064141 072040 060556 020110 050115 040511
 00154: 046040 066545 071563 060547 062440 066157 067553 071440 066151 065545 020151 067040
 00170: 043123 041440 040522 050101 020040 020040 020040 020040 020040 020040 020040 020040 020040 020040

M7084804 MAILDB RECORD 14 (%16, %E)

00000: 063157 071155 060584 026040 060556 062040 064556 020105 043124 020111 067164 062562
00014: 067141 066040 043157 071155 060564 027040 020040 020040 020040 020040 020040 020040
00030: SAME: TO 000110-1
00110: 020040 020040 020040 020040 020040 020040 020040 020040 040440 061154 060556 065440
00124: 066151 067145 020150 060563 020152 072563 072040 061145 062556 020151 067163 062562
00140: 072145 062054 020141 067144 020164 064151 071440 064563 020164 064145 020145 067144
00154: 020157 063040 072150 062440 066545 071563 060547 062456 020040 020040 020040 020040
00170: SAME: TO 000200-1

M7084804 MAILDB RECORD 0 (%0, %0)
00000: DCC 01LUKE
00074: SAME: TO 000200-1

M7084804 MAILDB RECORD 1 (%1, %1)
00000: SAME: TO 000200-1
00074: SAME: TO 000200-1

M7084804 MAILDB RECORD 2 (%2, %2)
00000: SAME: TO 000200-1
00074: SAME: TO 000200-1

M7084804 MAILDB RECORD 3 (%3, %3)
00000: Example of FSC and EFT formats
00074: LUKE 01Example of FSC and EFT formats / 8 . ATKINSON
00170: SAME: TO 000200-1

M7084804 MAILDB RECORD 4 (%4, %4)
00000: DISTRIBUTION
00074: LUKE 01 / 8 . ATKINSON
00170: SAME: TO 000200-1

M7084804 MAILDB RECORD 5 (%5, %5)
00000: SOLDPASSOFFICE 0
00074: SAME: TO 000200-1

M7084804 MAILDB RECORD 6 (%6, %6)
00000: ATKINSON, TERRY DCC 01
00074: SAME: TO 000200-1

M7084804 MAILDB RECORD 7 (%7, %7)
00000: Example of FSC and EFT formats
00074: LUKE 01 / 8 . ATKINSON
00170: SAME: TO 000200-1

```

M7084804.MAILDB RECORD 8 (X10, 8B)
0000 $OLDPASS.
00074 SAME: TO 000200-1
.....
M7084804.MAILDB RECORD 9 (X11, 89)
0000
00074
00170: SAME: TO 000200-1
.....
M7084804.MAILDB RECORD 10 (X12, 8A)
00000: SAME: TO 00074-1
00074: of FSC and EFT Message.
00170:
.....
M7084804.MAILDB RECORD 11 (X13, 8B)
00000: formats
00074
00170: SAME: TO 000200-1
.....
M7084804.MAILDB RECORD 12 (X14, 8C)
00000
00074 rry ATKINSON / LUKE/01
00170: SAME: TO 000200-1
.....
M7084804.MAILDB RECORD 13 (X15, 8D)
00000:
00074: FSC ARPA
00170:
.....
M7084804.MAILDB RECORD 14 (X16, 8E)
00000: format, and in EFT Internal Format.
00074:
00170: SAME: TO 000200-1
.....
X-HPDESK-PRIORITY: 3
X-HPDESK-ID: 10440217 0 0 0 "LUKE 01"
.....
Dated: 06/05/86 at 1603.
Subject: Example
.....
Sender: Terry ATKINSON / DCC/01
Part 1.
Contents: 2.
.....
TO: Te
.....
This is an example to show what an HPMAIL message looks like in
.....
A blank line has just been inserted, and this is the end of the message.
.....

```


Interprocess Communication Using MPE Message Files

Lars Borresen
Hewlett-Packard Company
Computer Systems Division
19111 Pruneridge Avenue
Cupertino, CA 95014

I. What is IPC?

Interprocess Communication (IPC) is a facility which permits multiple processes to pass information between one another. This allows large tasks that have been broken into independent processes, to synchronize their actions and exchange data. IPC can be useful in solving a variety of problems.

Large processes can be constrained by MPE's 32KB maximum stack size. One possible solution is to divide them into several smaller processes. These processes can be specialized to perform one specific function or a group of functions. Then each process fits into its own 32KB stack.

Some programs must handle conflicting requirements. These requirements may include performing lengthy calculations as well as dealing with transactions that require frequent service. If these programs are broken up into several processes, one process can be dedicated to monitoring the transactions, ensuring they get the constant attention they require. Other processes can perform other CPU-consuming functions. These processes can coordinate their efforts with the use of the IPC facility.

Some tasks require more processor time than is available on a single machine. These tasks can be divided into several processes and spread across multiple machines in a network. IPC can be used to pass data to different processes on remote machines. This division of processes across machines also allows the other benefits of distributed data processing.

Reliability is improved because the processes must interface through well-defined IPC records. Large data structures become resources that are managed by specialized processes. Other processes request or update the data with a set of special commands passed through IPC. Unauthorized access or unintentional corruption can be closely controlled.

When tasks are divided into independent processes, testing them becomes easier. Inputs and outputs to the various processes become IPC records, and tend to become well-defined commands and responses. The processes can be tested individually because editors can be used to build the input records. The output can be easily checked or redirected to a terminal or printer.

Finally, large programs may be implemented more quickly. The overall task can be divided into small pieces. Each programmer can work on a separate piece, and run it as an independent process. The development of the pieces can occur in parallel. The pieces are modular, and fit together through well-defined interfaces using IPC.

II. How does MPE implement IPC?

There are several ways processes can communicate under MPE. The most powerful is the facility provided by the file system, which uses a FIFO queue structure. Sending processes

can queue multiple messages that are stored until a receiver reads them, even across system SHUTDOWNs. Receiving processes are allowed to wait for messages on one or more empty queues. Messages are deleted from the queue when they are read.

The cooperating processes using IPC do not need to be related. They can even be running on different machines in a network.

IPC has several advantages because it is part of the file system. Most functions can be performed with standard file system intrinsics that programmers are already familiar with. The cooperating processes find each other through a known file name rather than having to determine each other's process ID. There are several different ways to perform I/O. The :FILE command can be used to redirect the I/O to another device, or change the way in which the message file is accessed. The existing file system security features can also be used.

The heart of IPC is the "message file". Message files reside partly in memory and partly on disc. MPE uses the memory-resident part as much as possible, to achieve the best performance. The disc-resident part of the message file is used only as secondary storage when the memory-resident part overflows. For many users of IPC, MPE never accesses the disc-resident message file.

III. What about MPE/XL?

The interface a user of IPC would see is identical for MPE/V and MPE/XL. In fact, they both use the same code. The only changes that are visible are concerned with the way extents are managed on disc. Using IPC does not hinder any efforts to move software to an MPE/XL machine.

IV. How do you use it? - A simple case.

IPC can be relatively easy to set up. Let us say that there is a large programming task. For one or more of the reasons above, it is to be divided into two processes. One process will interface with the user. This process will be referred to as the "supervisor" process. It does some processing tasks itself, and offloads others to another process. This other process, the "server" process, only handles requests from the supervisor and returns the results. The following is a description of how to set up the communications between these two processes. The terminology used is from SPL, but other programming languages can be used for most of the features discussed here.

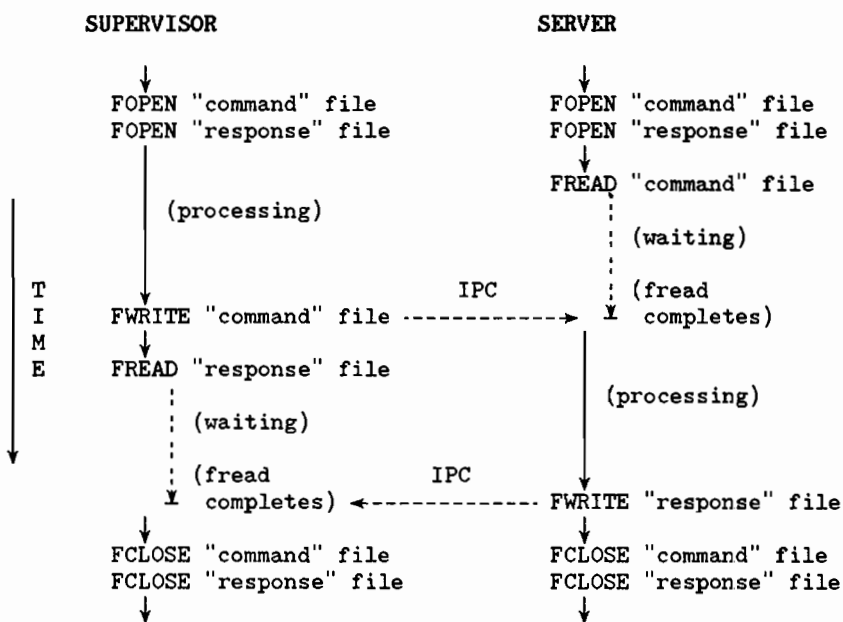
Program Structure

Like most other files, the message files need to be opened explicitly with the FOPEN intrinsic. Two-way communication is needed, so each process opens two files, one for supervisor-to-server commands, and the other for server-to-supervisor responses. The supervisor opens the command file with Write-Only access and the response file with Read-Only access. The server opens the command file with Read-Only access and the response file with Write-Only access. The FOPEN parameters are similar to any other FOPEN these processes would perform on another file. The processes make contact by using file names known to both of them. Assume the file names are always fixed and determined before the code is written.

In addition to FOPEN, these processes use three other file system intrinsics to perform IPC: FREAD, FWRITE, and FCLOSE. These intrinsics are used exactly as they would for any other type of file.

When the server starts executing, it performs an FREAD on the command file. The command file is empty, so the process suspends in FREAD instead of getting an End-Of-File condition. As part of the supervisor's processing, it eventually does an FWRITE to the command file. There is nothing to stop this FWRITE, so it completes almost immediately. The supervisor then does an FREAD on the response file, which, since it is empty, causes the process to suspend.

MPE notices that data has been written to the command file, and that the server process is waiting in FREAD for data from that file. MPE moves the data to the address that the server passed in the FREAD intrinsic, deletes the record from the message file, and restarts that process. The server picks up from the FREAD and processes the command. When it has finished processing the command, the server FWRITES its answer to the response file. At this point the server is done. It can terminate or issue another FREAD on the command file and start the sequence over again. MPE moves the response data to the supervisor's buffer, deletes it from the file, and restarts the supervisor at the FREAD. The supervisor continues processing, possibly repeating the cycle. Eventually both processes FCLOSE each of their files as part of terminating.



"Simple Case" Example

Boundary Conditions

Parts of this example merit some additional attention. One area is the "boundary conditions" which may prevent an I/O from completing: an FREAD from an empty file, or an FWRITE to a full file. The I/O intrinsic may return an End-Of-File indication, or it may wait until the condition is resolved (a record becomes available for the FREAD, or space in the file becomes available for the FWRITE).

The intrinsic waits if any of the following three conditions are true: there is an "opposite accessor" with the file open, this is the first I/O on this file this accessor has done since the file was opened, or this accessor is in "extended wait" mode. The reasoning is as follows: If there is an opposite accessor (if this process is doing FREADs and there is at least one process doing FWRITES to the file), then it is possible that the I/O request will be satisfied eventually. If there were no "opposite accessors", then the process might wait forever.

When processes start up, there may be a race condition in which it is difficult to predict whether one process's I/O request will be made before another process's FOPEN. For this reason, the first I/O waits to give the "opposite accessor" time to open the file.

Finally, a process may always want to wait rather than receive an EOF. It can do this by requesting "extended wait". This will be discussed in more detail later.

In the example, the server can open the the command file and issue the FREAD before the supervisor opens the command file. The FREAD waits because it is the first I/O after the open. But, if after the first command is processed the server issues another FREAD, and the supervisor has terminated unexpectedly, this FREAD will receive an EOF, signaling that something is wrong. This FREAD receives an EOF because the FREAD is not the first I/O, and there are now no writers accessing this file.

An FCLOSE by another process can cause an intrinsic that is waiting on a boundary condition to stop waiting and receive an EOF. If the process that is waiting is not using "extended wait", and the last process that can resolve that boundary condition does an FCLOSE on the file, MPE will wake up the waiting intrinsic and return an EOF condition.

Message File Names

In order to use IPC, processes must find each other through a known file name. This means that the file must be in a directory that is visible to all accessors. In the example, either a :BUILD is done previously, or one process must first FOPEN the file as new and FCLOSE it as permanent, to put the file in the directory. The file can be created as temporary if all accessors are running under the same job or session. If security is an issue, the file can be created with a lockword.

Recovery From Abnormal Terminations

In the event of an abnormal termination of the processes involved in IPC, some unread records may be left in the message file. Some applications are concerned only with current data and do not want to see unprocessed data from a previous run. The simplest solution is always to programmatically :PURGE and :BUILD all message files as part of the initialization processing. Another possibility requires the writer always to open the file first. If the writer's FOPEN has an Access Type (AOPTION 12:4) of Write-Only (=1) then the records will be automatically purged. If that FOPEN has an Access Type of Append (=3) then the old records will be kept and the new records will be added to the end. Note that if the reader opens the file first, then regardless of the writer's Access Type, the records will be kept. To cleanly recover the data in an application in which the reader

opens the file first, FLOCK the file so that the writer will not alter it, then FREAD all the records and process or discard them.

Message File FOPEN Parameters

Some things are different about message file FOPEN parameters. To open a new message file, the message file type must be specified. Either the FOPEN File Type field (FOPTION 2:3) must have the value for message files (=6), or the FOPEN must reference a :FILE command containing the keyword MSG. In the Exclusive Access field (AOPTION 8:2), the values 0 and 1 mean that only one reader and one writer are allowed to access this file concurrently (EXC in the :FILE command). The value 2 means that one reader and multiple writers can access this file concurrently (SEMI in :FILE), and the value 3 means that multiple readers and multiple writers can access this file concurrently (SHR in :FILE). The Access Type field (AOPTION 12:4) can have only the values Read-Only (=0), Write-Only (=1), and Append (=3). Execute, Read/Write, and Update access are not supported. If Read/Write access is needed to allow a process to queue a message to itself, this process should open the file twice, once Read-Only and once Write-Only.

Overriding FOPEN Parameters

Some values for FOPEN parameters are inconsistent with message files. When these values are used, MPE changes the value to one more in line with message files, rather than returning an error. An example is the No-Buffers option (AOPTION 7:1). Message files reside primarily in the buffers. Disallowing the file to be buffered would defeat many of the features of IPC. The bit is therefore changed so that message files are always Buffered. Related to this is the Number-Of-Buffers field. IPC needs to have at least two buffers. If less than two are specified, the field is changed to two. Also, IPC will never use more buffers than there are blocks in the file. If more are specified, the lower number is used. Since IPC is record oriented, the Multi-Record option (AOPTION 11:1) is always turned off. Finally, message files always use the Variable length record format internally. If Fixed length records are requested (FOPTION 8:2), IPC makes the records "look" Fixed, but a :LISTF will show that they are Variable internally.

To summarize, in the example four standard file system intrinsics are used to perform IPC: FOPEN, FCLOSE, FREAD, and FWRITE. FREADs wait for records, and FWRITEs wait for space, rather than returning an EOF, if a process exists which can satisfy the requests. FOPENs of message files are similar to FOPENs for other files. There are some parameters however, that are handled differently. Because the processes make contact using the message file name, the file must be in a directory that its accessors can use. Finally, applications may want to perform some recovery if they find a message file containing unread data from a previous run.

V. How do you use it? - Some variations.

Many users of IPC require more sophisticated functions than were described above. For these users, IPC provides several optional features that can be requested as needed.

Multiple Concurrent Readers or Writers

An application may need to use multiple concurrent readers or writers. In the example above, there may be multiple supervisors that need the services of the same server. The server may not be able to keep up with the requests, and so multiple copies of it may be needed to handle the command queue. Having a message file which is accessed by multiple concurrent readers or multiple concurrent writers sounds easy enough, but there are some

subtleties the user should be aware of. Each reader receives its own record when it does an FREAD. There is no "broadcast" facility in IPC, and hence no way in which all readers can receive copies of the same record. If either readers or writers are waiting on a boundary condition, they will be awakened in the order in which they called their I/O intrinsics. If both multiple readers and multiple writers exist, while a common command queue works well, perhaps individual response queues are needed. That way each supervisor only "sees" the responses to its commands. Often applications with this situation pass the name of the response message file along with the command, so that the file can be FOPENed and the response sent to the correct destination.

Preventing Deadlocks

It is possible, when multiple writers access a message file having a small FILE LIMIT, to cause a deadlock. MPE keeps track of when writers open and close the message file, by adding some extra records that are normally invisible. To leave room for these records, MPE increases the FILE LIMIT by 2 when the file is first created. This leaves enough room for one open and one close record *per file*. As writers open the file and begin writing, they each use one record for an "open" record and reserve another for their "close" record. One record is used and the space for another is reserved *per writer*. If the file has a very small FILE LIMIT, or there is a large number of writers, it is possible to get into a state in which all the space in the file is reserved. Writers are not able to write, because the file "looks" full (no available space). Readers are not able to read, because the file "looks" empty (no real records to be read). A deadlock has occurred. The best solution is to make the FILE LIMIT large enough that there is room for more records than there will be writers. There is another way to recover from a deadlock, which is discussed later.

Writer Identification

Sometimes it is useful for a reader to know when a new writer opens the file, which records this writer writes, and when the writer closes the file and hence no longer adds records. This allows the reader to keep track of who is sending records. It also may help the reader manage its resources, perhaps allocating a data segment when a new writer opens, and releasing it when the writer closes the file.

IPC provides this feature through the use of FCONTROL(file-number,46,parameter) (referred to as FCONTROL 46). Readers who call FCONTROL 46 receive two additional words at the beginning of each record returned by FREAD. The first word contains the record type (0 = data, 1 = open, 2 = close). The second word contains a writer's ID. The ID is only a way to associate the open, close, and data writes done by a specific writer. If additional information is needed (such as process ID number or program name), it can be passed by the writer in the first record, and saved by the reader in a table indexed according to the writer's ID. The writer's ID is always written when a writer does an FWRITE. FCONTROL 46 only makes it visible to the reader. Writers who call FCONTROL 46 will receive an error.

Extended Wait

If a process calls an intrinsic to perform an I/O, and it is blocked because of a boundary condition, then an EOF condition code is returned if there are no processes with this file open that can resolve the boundary condition. There are applications where it is useful for the process always to wait instead of receiving an EOF. It may be known that another process will eventually open the file and satisfy the blocked I/O request.

MPE lets the process request this "extended wait" mode through FCONTROL 45. "Extended wait" always starts out disabled, but both readers and writers can enable or

disable it. The rule is that the intrinsic waits on a boundary condition if any of the following are true: this is the first I/O, an accessor who can resolve the boundary condition has the file open, or the I/O is requested when "extended wait" mode is enabled. Calling FCONTROL 45 to disable "extended wait" does *not* mean that this process now never waits on a boundary condition, only that it will not always wait.

Timeouts

Suppose one process is using "extended wait", and, while it is waiting, the only process that could resolve the condition terminates unexpectedly. The waiting process will never wake up and must be ABORTed. It would be helpful if the waiting process could detect the situation and terminate gracefully. MPE allows the process to set a "timeout", so that if after the specified time the I/O still has not completed, the I/O intrinsic returns to the process. If there is a complex set of processes and message files, and they get into a state where all the processes are waiting on FREADs and none of them can do an FWRITE, timeouts can help detect this deadlock. Timeouts can be useful when a process must perform some time-sensitive processing (such as updating a table every N seconds), and therefore cannot wait for long periods of time. Some tasks are both command and time driven (display status every N seconds or when asked) and therefore could use timeouts.

Timeouts are set (in seconds) with the FCONTROL 4 intrinsic, and are valid for both readers and writers. Note that timeouts come into play only on boundary conditions. If an FREAD or FWRITE times out, a CCL condition code will be returned and FCHECK will return FSERR 22, "Software Time-Out". Currently timeouts on terminals are valid only for the next I/O, while timeouts on message files stay in effect on every I/O until explicitly turned off. We have had several requests to change message files to work like terminals, and are currently investigating it.

Non-Destructive Reads

Another problem occurs when a server is not able to honor a request (contained in a message file record) immediately because of a lack of resources. The requests must be handled in order, so the server tries to free the resources it requires. Eventually the server comes back and tries again to process the request. In this case it would be useful to have a "non-destructive" read, to look at the request record in the message file without deleting it. A test is made to see if the request can be satisfied. If it can, then a regular destructive read is done and the request is processed. If the request cannot be satisfied, it stays on the top of the queue, where it can be read and tested at a later time. This feature is provided by FCONTROL 47. It is valid for readers only, and is in effect only for the following FREAD. Note that repeated "non-destructive" FREADs always read the same record.

Forcing Records To Disc

As stated before, message files reside primarily in buffers. This means that they will probably lose data in the event of a system crash. For applications that must use message files and must be "crash proof", MPE provides a way to force the data to disc. This is done by using FCONTROL 6. Either readers or writers can call this intrinsic. It forces all buffers to be written to disc and the disc file label to be updated. This causes several disc I/Os and therefore takes a relatively long time. It must be done after each FWRITE to be most effective, and even then there is a "window" between the FWRITE and the FCONTROL 6 during which a crash would cause the record to be lost.

Remember that IPC is designed to be a fast, efficient means to pass messages between processes. If the task the application is performing is really event logging, and it must be

highly crash resistant, perhaps "circular files" or other files that can be used with FSETMODE would be a better vehicle.

Release G.01.04 (T delta 4) of MPE (and subsequent releases) contained several improvements that make message files more resilient to system crashes.

In summary, IPC provides some special features for those applications that can use them. These features include handling multiple concurrent readers and writers to a message file, and identifying the writer of a record. IPC also allows processes to always wait rather than receive an EOF on a boundary condition. It allows timeouts to be set on I/O requests, allows non-destructive reads, and permits the user to force all the buffers to be written to disc.

VI. How do you use it? - No Wait I/O.

Sometimes a programmer wants an application to read or write a record but does not want it to wait for an I/O to complete. For this application, waiting is "wasting" time when it could be doing other processing. Timeouts do not adequately address this problem. The programmer wants this application to start an I/O, continue processing immediately, and check periodically to see if the I/O has finished.

MPE provides a way to solve this problem with No Wait I/O. This feature is requested by setting the No Wait I/O flag (AOPTION 4:1) in FOPEN. When using No Wait I/O, the process must make at least two intrinsic calls to perform the I/O, one to start it and one to finish it. MPE still handles the file in the same way, but instead of waiting for the I/O to complete, MPE returns control to the application so that the application can do some useful processing.

No Wait I/O has been available to users of standard files for a long time. But to use it on standard files requires privileged mode, because on standard files the mechanics of No Wait I/O prevent MPE from protecting a process from corrupting its own stack. However, because message files work differently, No Wait I/O on message files does *not* require privileged mode.

No Wait I/O Intrinsic

To perform a No Wait I/O, the FREAD or FWRITE intrinsic must be called to initiate the transfer. These intrinsics return immediately, and no data is transferred yet. The return value for FREAD is set to zero, and is not needed. To complete the transfer, either IODONTWAIT or IOWAIT must be called. IODONTWAIT tests whether the I/O has finished. If it has, the intrinsic returns a condition code of CCE and the file number as the return value. If the I/O has not completed, CCE and a zero return value are passed back. If IOWAIT is called, it waits until the I/O has finished, like a normal Wait I/O FREAD or FWRITE.

Only one No Wait I/O may be outstanding against a file by a particular accessor at a time. However, accessors may have No Wait I/Os outstanding against several files at the same time. These I/Os may be completed by a "generalized" IODONTWAIT or IOWAIT (the file number parameter is zero or is omitted). In this case, these intrinsics report on the first I/O to complete, returning the file number for that file. If the call to one of these intrinsics is in a loop, then that one call can be used to complete all the No Wait I/Os.

Prior to MPE version G.02.00, the Target parameter was required in IODONTWAIT and IOWAIT when completing FREADs on message files. This was because the actual data

transfer to the process's stack takes place during these intrinsics, and there was a problem "remembering" the Target passed to FREAD. Starting with G.02.00 the Target parameter is optional, as it is for standard files, however not specifying it requires privileged mode for the same reasons as standard files.

Aborting No Wait I/O

There are cases in which, after a No Wait I/O has been started, something happens causing it not to be needed. MPE lets the process abort No Wait I/Os that have not yet completed by using FCONTROL 43. A condition code of CCE is returned if the I/O was aborted. CCG is returned if the I/O has already completed, and IODONTWAIT or IOWAIT must be called to clear it. CCL and FSERR 79 "No No-Wait I/O pending for special file" are returned if there was nothing to abort.

Currently, MPE does not support No Wait I/O to message files across a network. In many cases this is not an important limitation, because it is rare that both readers and writers to the same message file need to use No Wait I/O. If the file is made local to the accessor that needs No Wait I/O, the other accessor can then do Wait I/O across the network.

VII. How do you use it? - Software Interrupts.

No Wait I/O requires the application to "poll" to see if the requested I/O has completed. Each time the check is made, there is a certain amount of overhead, whether the I/O has completed or not. The application is faced with a hard trade-off. The more often it polls, the greater the overhead, and the poorer its overall performance becomes. If it polls less frequently, then the longer the delay between when the I/O can complete and when the application completes it, and so the poorer its performance when handling that message. One solution is to use software interrupts.

Software interrupts are most often used to handle messages from a high priority input queue, while time-consuming processing is done in the background. Perhaps a process is copying a large file across a network. The process posts an FREAD against its command message file. It then devotes all its time to performing the copy. If a high priority command is issued (requesting the number of records copied so far, or to stop the copy immediately), MPE causes the application to interrupt the copy, and forces execution of the application's interrupt handling procedure. There the FREAD is completed and the command is processed. When the interrupt handling procedure is exited, the copy is resumed automatically at the statement where it left off.

Software interrupts are really just a special case of No Wait I/O. The difference is that instead of the process polling to see whether the I/O can be completed, MPE interrupts the process when the I/O can be completed. Most of the discussion about No Wait I/O also applies to software interrupts. Like No Wait I/O, a call to IOWAIT or IODONTWAIT is needed to complete an I/O request.

Three intrinsics are specific to software interrupts: FCONTROL 48, FINTSTATE, and FINTEXT. FCONTROL 48 arms software interrupts for a particular file. It is also the way the application tells MPE the address of the application's interrupt handler. FINTSTATE is used by the application to enable or disable software interrupts for all files with interrupts armed by this process. It returns an indication whether software interrupts were enabled or disabled before this intrinsic was invoked. FINTEXT is used to return from the interrupt handler and leave software interrupts enabled or disabled.

There seems to be some confusion on how to use software interrupts. The following describes exactly how they are set up and used.

Software Interrupt Initialization

As with most other files, the message file must be explicitly FOPENed. Software interrupts are usually used when reading from the file, but there is nothing to stop a process from using them when writing. At this point the application can perform normal Wait or No Wait I/O on the file. When it decides to begin software interrupt operation, it calls FCONTROL 48, passing the PLABEL of its interrupt handler. The PLABEL is the address of the procedure and can be found by the SPL construct "label := @procedure-name". FCONTROL 48 returns the previous value of the PLABEL. Zero means that software interrupts were not armed.

At this point, the process can start the I/O (in this example, assume it is an FREAD). If the FREAD intrinsic was called before the FCONTROL 48, it would have been handled as a normal, non-software interrupt FREAD. FCONTROL 48 overrides the FOPEN No Wait I/O flag (AOPTION 4:1). Regardless of the setting of this bit, an IODONTWAIT or IOWAIT is needed after a software interrupt to complete any I/O started after the FCONTROL 48. A call to IODONTWAIT or IOWAIT before the interrupt occurs does not complete the I/O. MPE starts out with software interrupts disabled. If the FREAD is satisfied, the software interrupt is postponed until interrupts are explicitly enabled. The process uses FINTSTATE at this point to enable software interrupts for all "armed" files opened by this process. The call to FINTSTATE can occur anywhere in this sequence, but the other intrinsic calls should be made in the order given.

Interrupt Handler

The interrupt handler is a special procedure of the process, devoted to completing the I/O request after an interrupt occurs. This procedure is never called explicitly. Instead, MPE forces the process there when a software interrupt occurs, possibly from the middle of a statement. The procedure declaration can be either parameter-less, or have a single INTEGER parameter to contain the file number. This can be useful if there are several files using software interrupts, and some require special handling. Perhaps each file has a different buffer address passed to IOWAIT.

MPE automatically disables software interrupts and control-Y traps when it jumps to the interrupt handler. A call to IODONTWAIT or IOWAIT is needed to finish the I/O request. The I/O request can be completed immediately, so IOWAIT and IODONTWAIT will work the same way (IOWAIT will not wait). Check the condition code (a good idea after any intrinsic call). A CCG means the interrupt occurred because there is an End-Of-File condition. If a CCE is returned, the process has a record. It can be processed here, or a flag can be set to indicate that the record has been received and should be handled during the normal non-interrupt processing. Often, at this point, the next software interrupt is set up by performing another FREAD. The FREAD does not have to be done here, and could be performed elsewhere during normal processing. The last statement in the interrupt handler should be a call to FINTEXT. This allows the process to pick up where it left off when the interrupt occurred, enable control-Y traps, and optionally leave software interrupts enabled or disabled. Exiting with software interrupts enabled is usual, but the process may leave them disabled if the record needs special processing and it does not want any additional interrupts until it is completed. At that time it needs to call FINTSTATE to enable interrupts.

Main Line Code

For the most part, the "main line" code of a process does not need to be concerned with the I/O to message files using software interrupts. As long as interrupts are enabled, they can occur anywhere in user code. If one occurs during an MPE intrinsic, it is usually postponed until user code is re-entered. There are some exceptions. Interrupts can occur during a "generalized" IOWAIT, during an IOWAIT on another message file not using software interrupts, or during PAUSE and PAUSEX. PAUSEX is an alternate entry point into PAUSE. If PAUSE is interrupted part way through, when it is restarted it starts from the beginning of its timeout. PAUSEX, however, resumes where it left off, and only waits the remaining time.

The use of software interrupts introduces the possibility of a problem that applications normally do not have to think about. Some code is sensitive to interrupts. It usually involves data that is altered by both the interrupt handler and the "main line" code. For example, suppose the "main line" code decrements a counter and the interrupt handler increments the same counter. The "main line" code loads the old value and subtracts one from it. Before it is stored back, an interrupt occurs. The interrupt handler loads the old value, increments it, and stores the new value back. The "main line" code resumes, storing its new value on top of the interrupt handler's new value, and the increment is lost. One solution is to protect sensitive code by using FINTSTATE(FALSE) to disable interrupts before the operations, and FINTSTATE(TRUE) to enable interrupts afterwards.

Disarming Software Interrupts

It is possible to shutdown software interrupt operation and resume normal Wait or No Wait I/O on the message file. If there was an I/O posted against the file, you need to use FCONTROL 43 to abort it, just as in No Wait I/O. If software interrupts were disabled with FINTSTATE, the I/O completed, and the interrupt postponed, FCONTROL 43 returns a CCG. Interrupts need to be enabled to let the interrupt handler finish the request. Using FCONTROL 48, but passing a zero instead of the PLABEL, disarms the interrupt routines for the file.

There are some additional limitations. Currently, software interrupts are not available in COBOL, or on remote files. If the process contains privileged code, the interrupt handler must be privileged to handle interrupts from it, or else an ABORT 22, "Invalid stack marker", will occur.

To summarize, software interrupts are a way for MPE to notify a process that its No Wait I/O is ready to be completed. Software interrupts, however, require some special set up and a user written interrupt handler to complete the I/O. The I/O to a file can be hidden from the rest of the process. When not needed, software interrupts can be disarmed, and normal Wait or No Wait I/O can resume on the file.

VIII. Have there been any recent changes to IPC?

Many changes have been added recently to the G.01.XX (T-MIT) and G.02.XX (U-MIT) releases of MPE/V. If you are having problems with IPC, it is strongly recommended that you upgrade to a recent version.



IX. Summary

Interprocess Communication (IPC) can be used to solve a variety of problems that an application may encounter. The IPC provided by the File System is powerful and has many features that can be used by applications. If the task to be performed is simple, there is a simple way to use IPC to coordinate multiple processes. Four well known file system intrinsics are all that are needed. The use of Wait I/O further simplifies the programming. If the task has some special needs, IPC has several features that can help. These include writer's IDs, extended wait, timeouts, and non-destructive reads. If waiting for an I/O to complete is a problem for an application, No Wait I/O can be used to start the I/O, and periodically poll to see if it can be completed. Software interrupts provide a way for a process to start an I/O, and continue processing until the I/O is ready to be completed. At that time, MPE interrupts the process, forcing it to execute its interrupt handler, where the I/O is completed. Once done, execution resumes where it had left off. Finally, many improvements have been added to IPC in recent releases.



Brant Computer Services Ltd.

EXPERT SYSTEM MANAGER

Ross G. Hopmans

**Brant Computer Services Limited
Burlington, Ontario**

BURLINGTON
615 Brant Street
Burlington, Ontario
L7R 2G6
(416) 632-1386

TORONTO
6303 Airport Road
Suite 201
Mississauga, Ontario
L4V 1S2
(416) 673-9417

CAMBRIDGE
26 Colborne Street
Cambridge, Ontario
N1R 5S9
(519) 621-3233

EDMONTON
9637A 45 Avenue
Edmonton, Alberta
T6E 5Z8
(403) 438-9123

CALGARY
402-9203 Macleod Trail S
Calgary, Alberta
T2H 0M2
(403) 259-2482

VICTORIA
209-1095 McKenzie Ave
Victoria, British Columbia
V8P 2L5
(604) 727-6113

Toll Free 1 800 387-6704

ABSTRACT

We report here on the development of a new, rule based product which complements Hewlett-Packard's statement that an in-house data processing staff need not accompany an in-house computer. The product, an Expert System Manager, has been designed to run on a Hewlett-Packard HP3000 mini-computer and any PC compatible micro-computer and is meant to embody the knowledge of an experienced system manager to help an inexperienced user react when problems occur in the computer system and peripherals.

This paper explores the product definition, development and knowledge engineering.

INTRODUCTION

Brant is a supplier of computer software, services and support. The Expert System Manager was conceived for two reasons - to supplement our Facilities Management and Support offerings and as a documentation and educational tool to help us capture the knowledge of our in-house experts and to help train others.

The Expert System Manager was designed with an unsophisticated end-user in mind. The product must run on the micro-computer in case the HP3000 goes down and must converse with the user in a question and answer fashion. The system should lead to specific recommendations to remedy the problem with a minimum number of questions and should help the user restore the computer to an operational state as quickly as possible. The system should, in addition, explain its reasoning, address preventative maintenance and allow users to add to the base of knowledge in the system.

Rather than a strictly academic exercise, this product was conceived not only as saleable, but was seen as required in many situations. The Expert System Manager will be quite useful to new users of smaller systems who do not wish to hire specific data processing staff.

Large shops can benefit from the Expert System Manager to overcome problems which occur when the system manager is unavailable. Multiple site shops can use the Expert System Manager to maintain consistency across machines and sites. Since the base of knowledge is expandable, it can be tailored to the specific requirements and methodology where it is installed.

brant 

EXPERT SYSTEMS

"Expert" or "knowledge-based" systems are among the most exciting developments in the field of artificial intelligence. These programs embody the knowledge of a particular application area combined with inference mechanisms which enable the program to use this knowledge in problem-solving situations. Simply put, expert systems are machines that think and reason as an expert would in a particular domain.

The development of an expert system revolves around the ability to represent the knowledge and skill of an expert. Knowledge or facts are easy to represent. Conventional data bases are nothing but collections of facts and pointers. Building "skill" into expert systems presents more problems. Skill consists of a list of heuristics, the rules of thumb, that provide the "how-to" in problem solving. These rules free the computer from searching its entire data base for an answer to a problem.

An expert system must also contain the formulas and methods of reasoning that experts use in solving their problems. The task is to program the workings of an expert's mind and that falls to the knowledge engineers.

Expertise consists of knowledge about a particular domain, understanding of domain problems, and skill at solving some of these problems. Expertise usually involves more than definitions and facts, but includes rules of thumb called heuristics. Heuristics enable human experts to make educated guesses to recognize promising approaches to problems and to deal with erroneous or incomplete data. The central task of expert systems is to extract, record and reproduce such knowledge.

The emphasis is on knowledge rather than formal reasoning methods because most of the difficult and interesting problems resist precise description and rigorous analysis. In addition, human experts achieve outstanding performance because they are knowledgeable. If computer programs can embody and use this knowledge then they too should attain high levels of performance.

Expert systems are distinguished from conventional data processing by symbolic representation, symbolic inference and heuristic search. They differ from broad classes of AI in that they perform difficult tasks at expert levels of performance, they emphasize domain-specific, problem-solving strategies and they employ self-knowledge to reason, providing explanations or justifications for conclusions reached.



Knowledge engineering involves extraction, articulation and computerization of the expert's knowledge. Knowledge consists of descriptions, relationships and procedures in some domain of interest, and the engineer must first extract the expert's knowledge and then organize it in an effective implementation.

PROLOG

The two major development languages for implementation of expert systems are Lisp and Prolog. From the outset, this project was to be undertaken using MProlog, Logicware Inc.'s version of Prolog as the development language. The use of MProlog allows us to develop the system on the HP3000 and port it to the micro-computer or, as in this case develop the system on the micro and port it to the HP3000.

Selected as the basis for the Japanese Fifth Generation Computer System Project, Prolog enables designers to describe their applications in logical terms for interpretation by the computer. MProlog gives you a powerful inference engine surpassing the capabilities of most expert system shells. It provides automatic, system-driven reasoning with the rules and facts in the program knowledge base.

The substance of an MProlog program is a collection of facts and rules that the programmer creates. The facts and rules relate to a problem that the programmer wants to solve and MProlog provides a built-in inference mechanism that acts on the facts and rules.

The use of MProlog was confirmed by the fact that the data was reliable, well structured and that the decision process involved feedback and parallel decisions.

PRODUCT DEFINITION

It is important in any expert system to restrict the scope of expertise to a manageable size. System managers are responsible for solving numerous and varied problems, as we learned in the observation of our expert.

We decided to limit the scope of the Expert System Manager to the class of problems associated with returning a machine which has gone down to an operational state. This is a critical function of the system manager and one in which success and failure are easily determined.

The knowledge engineer conversed with the expert to

brant 

informally characterize the class of problems to solve. The major domains of expertise and the basic concepts, primitive relations and definitions were established before determining the domain of the Expert System Manager.

RESOURCES

To obtain the knowledge base necessary to our Expert System Manager, we used one of our own system managers as the expert.

A paradox of knowledge engineering is that the more competent domain experts become, the less they are able to describe the knowledge they use to solve problems. Domain experts need outside help to clarify their thinking. Hence, our expert was not also our knowledge engineer.

Our knowledge engineer was chosen as an articulate employee with a well rounded knowledge of the computer industry and the system manager's environment, but was not an expert in the field.

KNOWLEDGE ENGINEERING

Our goal in the knowledge acquisition was to transfer the problem solving expertise from our system manager to a computer program.

The knowledge engineer was not himself an expert in operations and so was keen to note the organizational mechanisms used by the expert to classify the type of problem. The organizational constructs form the basis for certain types of inferences the expert makes during problem-solving and constitutes the structural expertise about the domain.

On another level, the knowledge engineer listened for the basic strategies the expert used when performing the task. What facts did the expert try to establish first? What kinds of questions did the expert ask first? Did the expert make initial guesses about anything based on tentative information? How did the expert try to refine the guess? In what order did the expert pursue each of the subtasks and how did this order vary in case studies?

The strategies and structures couple to constitute the expert system's inference structure. In other words, we have established what tasks and terms to determine and how and when to apply them. For documentation purposes, the knowledge engineer must also listen for justifications the expert uses

brant 

when solving a problem.

Problem identification was undertaken by defining each problem identified, its aspects, characteristics and sub-problems. The objective was to characterize the supporting knowledge structure for each problem to begin building the knowledge base. Several iterations were required for each problem. Some problems were split into multiple problems if considered too large. The following points were considered important:

- . what class of problems does the expert solve
- . how are the problems defined
- . how are the problems partitioned
- . what are the important data
- . what situations are likely to impede solutions
- . what does a solution look like

Our knowledge engineer and expert, after several cycles of restatement, isolated the knowledge relevant to solving the problem and identified the key elements of the problem description.

It was important to us not to ask our expert directly about his rules or methods for solving a problem. In general, domain experts have great difficulty in expressing such rules. Experts have a tendency to state their conclusions and reasoning in general terms, too broadly for effective machine analysis. The expert makes complex judgements rapidly whereas the machine will operate at a more basic level.

We did not believe anything our expert said outright. Working hypotheses were developed based on information from the expert which were tested for validity and consistency before being accepted. The tests involved having the expert solve new problems using the hypotheses - the expert had to demonstrate the use of his rules during actual problem solving.

We found in undertaking our knowledge engineering that our expert used patterns to serve as his index to his store of information rather than simply a list of facts. The use of patterns is what is known as intuition.

Within his area of expertise, our expert quickly recognized new situations as instances of things with which he was already familiar. However, when he was faced with new situations, he applied general principles and deductive steps. It was more insightful to us to present the expert with novel problems to decompile his knowledge and view the actual problem-solving activity.



Several methods were employed in the knowledge acquisition. The first was observation. We simply listened and observed the expert in his problem-solving, without interfering or interrupting. It became necessary to record our observations and analyse the transcript afterward. These recordings were done at times by a knowledge engineer and at other times by an assistant to the expert during their problem solving sessions.

A second method was an attempt to have our expert record what he thought were the rules and processes he went through in his problem solving activity.

Our most successful method was a combination of the two - observation of the expert with his own analysis of what he was doing along with intervention by the knowledge engineer to clarify fuzzy points.

IMPLEMENTATION

We chose to have all questions answered simply as Yes or No. Further courses of action were contingent on the previous answer. The first question encountered by the user was

"Is there a problem with the computer system?"

An answer of No would cause the program to terminate whereas a Yes causes the program to go on to the next level and establish the thrust of the questioning.

"Does an error message appear on the console?"

If an error message appears on the console, the system will pursue questioning on the nature of the error and can establish the type of fault. For some problems, corrective action is recommended; for example, putting a drive back on-line if indicated. In most cases, however, the system will recommend that the Hewlett-Packard SE be called.

If no corrective action has been recommended to this point, the system will try to establish if the console is accessible.

"Does the colon prompt (:) appear on the console?"

This question establishes whether or not the operator can use the console at all. If the answer is No then the program pursues the avenue of establishing why the console may not be responding.

brant 

EXPERT SYSTEM MANAGER

7

"Can any other terminals access the computer?"

The program tries to localize the problem. If other terminals can access the computer then the system is obviously up and the operator will be led through a series of questions regarding configuration, cabling and connections.

If, however, no terminals are accessible then the program leads the user through checks of the activity lights and power supplies.

Our expert system manager may reach the point of suggesting that the user take the system down and will go through the steps of halting the system and bringing it back up.

In the worst case that nothing can be established by the program to cause the problem, the user will be directed to his or her SE and the program will suggest that the eventual knowledge be added to the knowledge base.

At any point the user can ask WHY for an explanation of the cause and effect relationship behind the question.

TESTING AND VALIDATION

For validation, the knowledge engineer picked a set of representative problems for discussion with the expert. The purpose was to determine how the expert organizes each problem, represents the concepts and handles inconsistent or inaccurate data.

The process of acquiring, implementing and testing the system involved our case studies, textbooks, operations manuals and other specialists to supplement the knowledge of the expert. As the knowledge base grew and was refined, it became an executable program.

Testing of the Expert System Manager involved evaluation of the system both in ease-of-use, completeness and correctness. A variety of examples were run through the system first with the case studies collected over a period of time and finally in conjunction with the expert in solving problems as they occurred.

Problems were chosen to fully challenge the system. The input/output system was continually refined as was the knowledge base. In some cases, the wrong questions were being asked and in others, insufficient information was being

brant 

gathered.

Used by someone with an intermediate level of experience, the Expert System Manager proved to be illustrative and educational. The trace facility of MProlog displays the rules fired to the user to show the reasoning process used to arrive at the final conclusion. This was invaluable in refining the system to determine not only where the rules were wrong or inconsistent, but why.

CONCLUSIONS

The prototype developed to this point has shown to us that we can indeed capture the knowledge of our experts in a tractable form. The knowledge is then available for use by the inexperienced and expert as well.

This will allow the knowledge of other experts to be added to the knowledge base so it can continually be enhanced and expanded.

It is in our plans to expand the domain of the expert system to more fully represent the expertise of the system manager.

brant 

THE ROLE OF THE OPERATOR IN THE DATA CENTER OF THE FUTURE

RONALD DRAKE
OPERATIONS CONTROL SYSTEMS
560 San Antonio Road
Palo Alto, California 94306

INTRODUCTION

The role of the operator in the modern data processing environment is an important one. The operator is the person in closest contact with the machine itself. The operator is often the source of first resort in the resolution of user problems. The operator is critical to the effectiveness of any disaster recovery program. In addition to all this, the operator is responsible for the timely completion of all scheduled data processing tasks. A productive data processing center depends on a productive operator. As computers have become more important in the modern world, so, the operator has become more important in keeping these tools functioning effectively and efficiently.

However, as advances are made in computers--advances in hardware, operating systems and software-- we are faced with the question of the role of the operator in the data center of the future. What will an operator do? What will an operator be? Perhaps, most importantly, will there be operators at all? This paper will address these questions.

BACKGROUND

Before the advent of modern multi-tasking, multi-processing machines, computer operations was largely the province of the engineers and programmers who developed them. A fairly large body of specialized knowledge was necessary to understand and operate older computers. Advances in microprocessor technology and the attendant development of the video display terminal, mass storage devices, and improvements in operating systems changed that.

| PREVIOUS | CURRENT |
|---------------------------|------------------------------|
| Engineers and Programmers | Computer Operators |
| Single Function | Multiple tasks and processes |
| Monitors | Operating Systems |
| Mainframes | Minicomputer and PC |
| Developer | Overseer |

Figure 1: Operator Characteristics

Microprocessors made computers accessible to business. They were faster, easier and cheaper to run. Mass storage devices like tape and disk drives allowed information to be stored compactly and cheaply. Microprocessor-driven access routines could fetch vast amounts of information and instructions and route them to various destinations with a high degree of accuracy making possible operating systems capable of accommodating hundreds of users.

Advances in programming languages made "user-friendly" applications possible. And the data these newly-empowered users generated could be propagated across an office or around the world. Because the second-generation computer allowed so many people to generate so much output and because it could operate on multiple sets of data and generate the results in batch mode, a human intermediary was needed to oversee the flood of information spawned by the computer. Someone was needed to tell the computer what to do and when to do it. That someone was the computer operator.

THE OPERATOR IN THE PRESENT

As we speak, billions of dollars are dependent on the diligence and expertise of computer operators. Critical facets of American business--financial transactions, sales orders, billing-- are under the control of the operator. The operator launches the batch jobs which process these transactions. A good operator is capable of performing job recovery in the event a batch job terminates abnormally. In addition to this important task, operators are responsible for the archiving of data. Operators maintain peripheral devices and replenish paper and tape when needed. The operator is the first on the scene when operation of the machine is interrupted for reasons other than regularly scheduled shut-downs and is often the person responsible for bringing an errant computer back on-line. Because operators are in attendance as long as the machine is running, they are depended on by users to answer questions about individual applications or the system itself.

These functions are of the utmost importance. The present-day operator must possess certain basic qualities if she or he is to be effective.

An operator must, first of all, understand the whys and wherefores of the job. While some knowledge of the principles of operation of the computer itself is helpful, an operator MUST know the full range of system commands. The operator must know how to interpret job-control language. Most of us have been faced with the prospect of re-running production because a critical early job ended abnormally, unbeknownst to the operator.

Because users turn to them for assistance and advice, an operator has to be a "people person"; that is, an operator should be able to intervene on behalf of the user in a courteous, cordial manner. An operator should be able to tell the user what is being done and why. The public relations garnered for the data processing department in this way can prove invaluable.

An operator must be trustworthy and conscientious. While this smacks of a "Boy Scout" mentality, it is, nonetheless, true. Operators are privy to the most sensitive aspects of corporate life. They run payroll checks. They know where personnel and medical records are stored. Because of their familiarity with account structure, they have a direct line to administrative data, manufacturing and production data, and other sensitive or classified information. Just as important, operators have custody of the machines themselves. The manager of a data center must be confident that the operator will do nothing to jeopardize this critical trust.

THE OPERATOR IN THE FUTURE

Given the importance of the present-day operator, what are the prospects for the future?

To answer this question we must, again, look to the prospective technical advances in data processing.

The most important changes we can anticipate are those in the machines themselves.

Microprocessor technology is what took the computer from the warehouse to the desk-top and, if the last twenty-five years are any indication, it will take us further. The ability to store vaster amounts of information on ever smaller media--the ability to retrieve that information at speeds unhindered by the need for a read/write head to find and fetch information--will change the way way we relate to the computer. The need for disk and tape drives will decrease as chips with greater storage capacities are pressed into service.

These new machines will possess the capacity for processing greater amounts of information than ever. This new power will become important as the relatively new discipline of Artificial Intelligence pushes back the barrier between the computer and the human mind. Expert systems are already in service which can accept a range of information and posit a diagnosis based on rules supplied by humans. But humans learned these rules by trial and error combined with a capacity for intuition and reasoning. Humans learn by practical experience in real time in the real world. The great challenge of Artificial Intelligence is the translation of this power to the computer. Once the greatest hurdle is passed--that of enabling the computer to assimilate and act on ambiguous, symbolic information (i.e. speech and pictures)--the barrier between the computer and the human mind will become much less imposing and, thus, easier for the average person to cross.

Artificial Intelligence will also expand the ability that some computers now have to diagnose hardware problems. A machine hardened against hardware failures and natural disaster would free the operator from tasks related to preserving the machine's state at the time of the failure and bringing it back into operation once the problem has been solved.

The hallmark of data processing in the future will be greater processing power distributed to greater numbers of people. The intelligent workstation will become more prevalent. The ability of users new to the computer to tell the computer what is required; the ability of that computer to accept, interpret and execute those instructions (and even make suggestions, based on its "knowledge" of the user); will de-centralize the computing environment. Modern data communications technology will transfer information that must be shared with other users.

The ability to verbally command the computer's power is not a speculation: it is here today. As this capability is refined, we can anticipate a day when users with no programming experience at all will be able to develop in minutes applications that take programmers months to code and test.

These developments will, of necessity, generate more independent, autonomous users. These users will have control of their own machines and peripherals. Archiving of data, now dependent on large electro-mechanical devices, bulky storage media and human intervention will be automated and internal to these individualized machines. Printing of documents may also be accomplished through the use of small, on-board printers or by transmission to a common printer.

There will be little or no need for the operator in this environment. The user, in conjunction with the intelligent processor, will be able to perform all the functions necessary for proper operation of the machine.

We see these inroads being made even in the infancy of Artificial Intelligence. As a programmer and former operator, my specific interest has been the automation of operations tasks on the HP3000. OCS/3000, the pre-eminent operations control application in the Hewlett-Packard processing environment, has optimized job-scheduling and monitoring by removing the necessity for an operator's intervention in the streaming of batch jobs. OCS/3000 obviates the slack time caused by a busy or inattentive operator thus making full use of the machine's resources. Complex time, date, run-book and pre-job dependencies are also considered automatically. It is easy to foresee the time, given the course of Artificial Intelligence research, when job recovery as well as job initiation can be automated. An expert system supplied with the nature of the error, the files involved and a prescribed course of action could effect such a recovery.

Computer manufacturers are increasing the ability of their machines to diagnose and report extraordinary conditions. Newer machines are able to carry out the system initiation dialogue with no operator intervention other than the resetting of the system's time-of-day clock.

In the final analysis, machines will become easier to use. Users will be free to perform for themselves a wide range of what are now operations tasks. The HP3000, in particular, lends itself to this development due to its powerful yet easy-to-understand commands and the ability to distill multiple commands into User-Defined Commands. A thorough system manager, an educated user population, a well organized system, and the installation of appropriate applications minimize the need for operator intervention. This is a desirable goal; one that can be accomplished in the present.

CONCLUSION

Operators are a critical and much undervalued asset in today's data processing environment. However, the need for the operator will decrease over time as machines, peripherals, and users become more sophisticated. Indeed, once the need to attend to printers and tape drives has been supplanted by advances in technology, operators may not be necessary at all.

Until that time, the operator will continue to be an indispensable member of the data processing community. Operators will play an important part in effecting the changes that are to come. They are the experts whose knowledge will drive the expert systems in the future. Operators will be crucial resources for users in the effort to understand and use the new machines to be developed between now and the end of the century. Operators should welcome rather than dread change. An attitude of resistance to advances like Artificial Intelligence and productivity tools like OCS/3000 is expected of those whose perspective is limited to the short-term. Those whose vision extends to the future can and will take advantage of the time gained to broaden their knowledge of the machine. Data center managers, analysts, and programmers have a vested interest in affording operators every opportunity to learn and grow. Operators with drive, intelligence and foresight will thrive in such an environment. More importantly, the users, the data center and the organization at large will benefit from the investment in time, training, and material support.

PROTECTING YOUR SOFTWARE INVESTMENT

BETSY LEIGHT
OPERATIONS CONTROL SYSTEMS
560 San Antonio Road
Palo Alto, California 94306

INTRODUCTION

Current software development and maintenance practices are expensive, difficult to manage and often lead to products of inconsistent quality. Recognizing these problems, we instituted a set of standards and procedures that go beyond ordinary measures to bring organization and control to the assortment of files and activities in our HP 3000 installation.

Before implementation of our current standards and automated tools, file management was a time consuming and often error prone activity. Informal procedures relied on the good intentions and sharp memories of either system users or overworked data center personnel acting as librarians. This paper will describe the selection and implementation of an approach to managing the organization of our files, users, and data processing procedures. The approach discussed is a complete system that standardizes operations and protects software and data investments.

NEEDS ANALYSIS

Initially, we performed a brief situation audit to determine the level and type of file management needed within our shop. First, we considered the value of our internally developed software. As a software development firm, we maintain years of development work. Furthermore, like most installations our system maintains several outside vendor purchased packages which add significantly to our software investment. Despite backup procedures, inadvertent changes to master or production files could cause production delays and errors. In some cases, these changes could even wipe out original copies of irreplaceable software. Second, we assessed the costs related to users working on the incorrect files. Generally, these costs appeared in the form of transferring the wrong software into production, reruns, faulty decisions based on faulty data, personnel overtime, poor response time and lower system throughput.

With these problems in mind, we identified several goals for our shop. We needed a file integrity approach that supported and enhanced existing operating system security. Additionally, our security approach had to maintain an orderly set of rules for all file transfers and include a logging capability on all violations. Our users needed to maintain convenient access to files, while at the same time, management wanted the valuable software data assets and production environment to remain secure. Finally, we needed a set of reporting capabilities that could tell our users where their files were, who had accessed or changed them, and what steps were necessary prior to placing each file into production.

SOFTWARE IDENTIFICATION

Initially we found that our software was scattered throughout several accounts. Our first step was to categorize files into different groupings such as source, object, libraries, jobstreams, UDC files and documentation. Second, a set of naming conventions was developed which considered the file, group, account and system, as well as the previously chosen file type. The operating system facilitated the organization of our software with standard account structures.

As in most shops, our programmers had adopted different conventions to identifying versions or releases. Our next step was to institute a standard approach to account for versions within our naming conventions. In some cases, releases consisted of collections of files that were made up of several more files, each with a variety of version numbers.

For example: Release 1.0 of Product A may consist of 75 files. The first 40 files are Version 1.0 and the remaining 35 files are Version 1.1.

Additional information was incorporated into file names to account for identification of the file's original name, its new name, family name and file set name, in addition to the version/release codes.

Our naming conventions allowed us to easily define how many generations of changes to retain for each file. Furthermore, we could quickly retrieve previous versions of a file and compare them to the most recent version on the system.

FILE MOVEMENT RULES

Here again, our first step was one of simply categorizing. In this case, we categorized system users into distinct groups such as creator, programmer, analyst, librarian, operator, quality assurance, production user and auditor. We then identified the most common types of file movements for each of these users as well as those moves which would be permissible.

The sequence of transfers was usually a series of moves, and required that the first move be completed successfully before the second could begin. To define the appropriate sequences, we created applications and defined the hierarchies and logical relationships of files within each application. These applications included groups of files that were logically or physically associated. Allowable file movements were defined in terms of steps that corresponded to standard procedures. The steps were then combined into routes that defined sequences of file movements for a set of files.

A simplified route appears in Figure 1. A file is checked out of its master location and moved into a development account for changes. The programmer then submits the file(s) to the Q.A. function. Management approval must be received prior to the check in step where the file goes back into production.

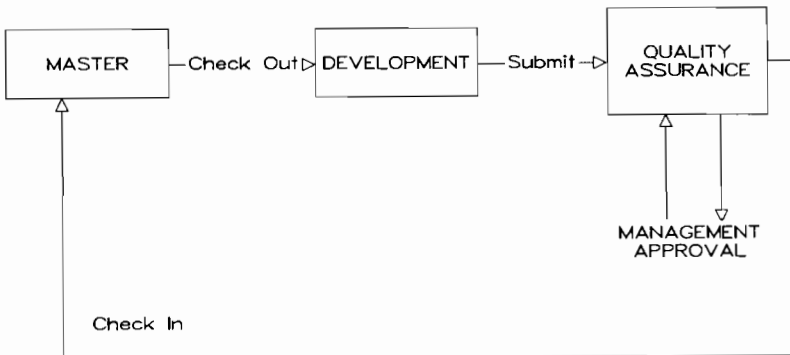


Figure 1: A Simplified File Route

At this point, we developed an automated tool called OCS/LIBRARIAN to facilitate the modification and definition of file organization. Initially the system aids in defining each sequence of file transfers. Once installed, OCS/LIBRARIAN ensures that every step is accounted for in the established development process. Code reviews, approvals, signoffs and controls are all enforced and tracked. As each transfer occurs, OCS/LIBRARIAN tracks the move and provides detailed audit records. Any attempts to transfer out of sequence are noted and attempted access by unauthorized users is also recorded. In the event of an emergency or exceptional situation, authorized users can move files out of sequence subject to the same tracking and reporting visibility as normal file movements. Users need only the seven simple commands outlined in Figure 2 to perform their file movement functions.

| FUNCTION | DESCRIPTION |
|----------|--|
| COPY | Duplicate a named file |
| PURGE | Delete a named file |
| MOVE | Duplicate a named file; Delete original file |
| APPROVE | Manual activity requirement |
| PERFORM | Online batch request to copy, move or APPROVE |
| RENAME | Change name of file |
| RELEASE | Modifies file access security |

Figure 2: OCS/LIBRARIAN FUNCTIONS

FILE ACCESS CONTROL

OCS/LIBRARIAN was designed using a file integrity approach that supports and enhances existing operating system security. All existing operating system access restrictions still apply, yet users maintain convenient access to needed files.

As do most shops, we utilized user logon identification and a set of user capabilities and file access matrices as part of our standard security. However, we needed clearer definition of WHO could access a particular piece of software. To achieve this, OCS/LIBRARIAN provides a global lock of the software investment and a single interface to all files. This file protection allows the system to intercept the multiple logon identifiers before any of the users (authorized or not) access any file. User logons can be grouped into a USER SET and then become part of the authorized group to access any particular software.

This approach allowed us to define rules for each file or group of files and to customize the level of security required for each one. Additionally, we restricted file moves and copies at each level to those specific individuals responsible for a particular set. Documenting which users were responsible for specific file movements and increasing accountability drastically reduced the amount of supervisory time required to manage files.

USING THE SYSTEM

Once the automated OCS/LIBRARIAN system had been developed, several reports were written to provide a management tool to track the progress of enhancements, problem corrections, new programs and other factors. The system tracks all defined file movement rules and file movements to create a history of all changes as well as a complete audit trail.

As a result, management can review changes as they take place and verify that they are properly authorized. Users and managers can determine where a given file is in its defined route as well as what has happened to a file in the past. This automatic file tracking allows users to determine who has which files. Thus, programmers always have an accurate record of which files they are currently working on.

Initial start up is simplified by an audit mode feature that can be loaded in an OBSERVE ONLY state. This provides a shop with a listing of system activities which give a clear picture of a particular installation's environment. Once this has been completed, an automated loading utility steps users through the process of defining applications and file organization.

CONCLUSION

System managers have long observed the rate at which hardware improvement has outpaced gains in personnel productivity. Unfortunately, implementing standards and controls to reduce errors and increase productivity has often forced the delay of critical development schedules. However, through a clear definition of one's software environment combined with the use of automated tools such as OCS/LIBRARIAN, a shop can automate its file management and file tracking process relatively easily. Today's data centers need the optimum environment for software development, testing, maintenance, and production. After all, isn't it about time to bring the power of mainframe EDP standards and a proven Library Control philosophy to the HP 3000?

AUTOCHANGER EXTENDS CARTRIDGE TAPE DRIVE CAPACITY

Manuel Escuder

Hewlett-Packard
Computer Peripherals Bristol Division
Bristol, ENGLAND

**Summary**

As the amount of on-line storage increases in mid-range systems with larger, non-removable discs, the issue of automatic backup becomes more significant. The 9144A, with its compact 1/4" cartridge, can provide system backup at very low cost provided the system can be dedicated to this task (eg overnight or at weekends). For systems whose total capacity exceed a single cartridge (67 Mbytes), operator intervention is required to perform media changes. The 35401A autochanger eliminates this operation by providing automated media changes, allowing unattended system backup to be scheduled during system off-time without operator intervention. This new type of CS-80 drive offers a large capacity increase while maintaining full media compatibility with earlier drives.

Introduction

A constant goal in the computer industry is to satisfy existing and future customers with products that offer competitive price and performance characteristics. Adding to this challenge, is the need to maintain compatibility with earlier devices and media in the case of mass storage peripherals.

For backup devices, there are two major performance parameters, transfer rate and media capacity. The traditional system backup method requires a dedicated operator to be present to manually load and unload tapes. This operation is usually performed outside normal system operating hours and operators have to be assigned to system backup after normal working hours, or the system has to shut-down early on pre-arranged dates.

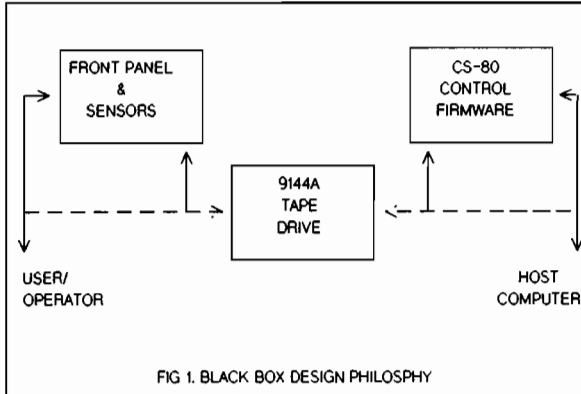
The introduction of 1/4" cartridges provided a very compact and easy to use media. A cartridge which is simply pushed into a "slot" is a far simpler proposition than threading tape in a 1/2" reel tape drive. Ease of use and low-cost has made these devices a very attractive alternative for small & mid-range multiuser systems where trained operators are not necessarily available. System backup in these conditions becomes even more of a chore. The transfer rate for 1/4" drives is considerably lower than that of a reel-to-reel tape drive and operator intervention outside system up-time is even more of a necessity because of the longer time needed to backup and the unacceptability of system down-time during working hours. Eliminating the need for operator intervention during system backup is clearly a very desirable benefit for these systems. An autochanger device can automate cartridge tape changes, providing a large increase in effective

capacity while maintaining media -and device- compatibility.

The 35401A autochanger combines the 9144A 1/4" cartridge tape drive with an autochanger mechanism. The device eliminates manual loading and unloading of tapes, allowing unattended system backup to be scheduled without operator intervention. The drive allows up to 8 standard 1/4" cartridges to be used in a purpose-built carrier or magazine. The magazine can be considered as a "new" single media consisting of up to 8 individually addressable cartridges and giving a maximum total capacity of 536 Mbytes. The drive is 100% media compatible with existing 1/4" cartridges and its operation is plug-compatible with the 9144A

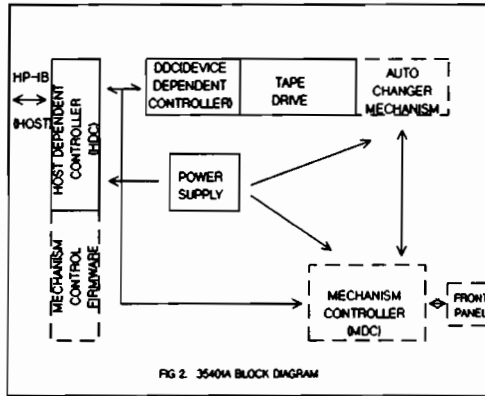
35401A Device Description

One of the key objectives in the design of the autochanger drive, was to maintain compatibility with the 9144A drive. This would ensure existing systems support could be leveraged, thus minimizing the amount of system integration effort while allowing a wide range of system users to start using the device as a natural extension of the 9144A drive. This goal was achieved due to the adoption of the "black box" design philosophy. This is shown in Fig 1 where the broken lines represent "normal" input/output as to a standard 9144A drive. In the autochanger environment, these paths are "broken" and re-routed via unique hardware and firmware. The operation of the 9144A within the 35401A is identical to its stand-alone version. Hardware and firmware input/output signals to the 9144A modules, are routed via the autochanger controls which coordinate activation and sequencing correctly in the modified environment.



Hardware and Mechanical Implementation

Figure 2 is a simplified block diagram showing the 35401A drive implementation. Broken line components represent unique autochanger modules added to the 9144A modules which are represented in solid lines.



The Host Dependent Controller (HDC) is a standard board used by several mass storage peripherals, including the 9144A. No hardware modifications have been made to it, but the resident microprocessor is time-shared with existing tasks and runs additional device firmware. This module provides the mechanism control and all high-level host interface tasks as well as the overall control and sequencing.

The Device Dependent Controller (DDC) is a standard 9144A module. It has not been modified in any way so as to allow direct use of the current production standard hardware and firmware. The HDC-resident control firmware is responsible for sequencing tape drive and autochanger mechanism operation.

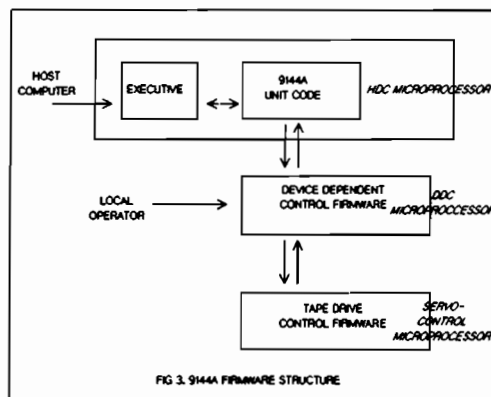
The tape drive mechanism and electronics are also a standard 9144A module. It has been subjected to minor mechanical modifications which can be performed in the field and consist of replacing some springs and fasteners. Sensor inputs normally received directly by this module have been re-routed through the mechanism controller to allow correct sequencing of its operation.

The Mechanism Dependent Controller (MDC) is unique to the 35401A. It interfaces to the HDC board using the same internal bus utilized to interconnect the HDC and DDC in a standard 9144A. The MDC performs all the sequencing and control tasks needed to move cartridges between the magazine and the 9144A tape deck. It requires no dedicated microprocessor as it is possible to time-share that of the HDC .

The autochanger mechanism provides the physical means to move cartridges between magazine and tape deck. It contains a number of sensors and actuators required to provide operation and position feedback to the controlling firmware and hardware. Essentially, the magazine and tape deck remain fixed. An elevator platform is capable of vertical motion. A horizontal pick arm assembly fixed to the elevator is capable of motion in the horizontal plane and can transfer cartridges between the platform and the magazine or between the 9144A tape deck and the platform. The mechanical design poses no restrictions on the order in which the cartridges are loaded into the tape deck, allowing "random" access to individual cartridges as directed by the user via the front panel or through the host computer.

Firmware Implementation

The standard 9144A firmware structure is shown in a simplified form in Figure 3. There is a hierarchy of tasks which directly relate to separate microprocessors. The highest level tasks (EXEC and UNIT) are resident in the Host Dependent Controller (HDC) processor. This is essentially a "round-robin" time-sliced operating system which performs the host computer interface control and the higher-level device control functions. Inter-task communication is performed by dedicated memory buffers (memos) which are controlled by the device operating system.



In the 35401A environment, all the 9144A tasks are treated as "black boxes". The EXEC has been modified and two new tasks introduced within the HDC firmware, as shown in figure 4. The combined effect of these 2 changes is to intercept EXEC memos -corresponding to host computer commands- and to sequence their execution, passing them onto the 9144A UNIT code for further action at the correct time for coordinated operation with the autochanger mechanism. For example, a host computer command to UNLOAD the current cartridge from the tape deck would be executed as follows

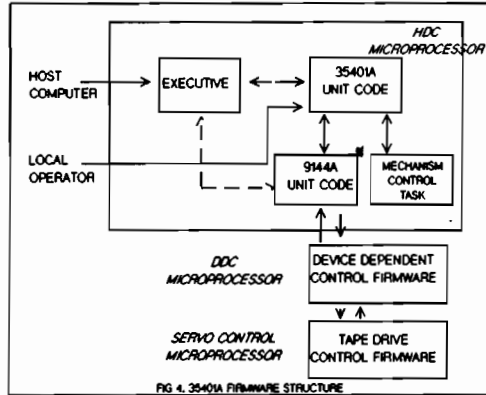
- The EXEC decodes the request and generates a memo to the 35401A unit code.
- The memo is immediately passed on to the 9144A unit code. This

firmware now generates the same sequence of operations normally executed by the stand-alone 9144A when unloading cartridges. It involves a tape rewind and use log information updates.

- Upon completion of the 9144A unload, the mechanism control task is informed. This code now generates the autochanger control sequences needed to retrieve the cartridge from the 9144A tape deck and replace it into the cartridge magazine.

- The mechanism control task informs the 35401A unit code task of Unload operation completion.

- The 35401A Unit code task then generates a command complete memo which is passed to the EXEC indicating completion.



This approach is conceptually the same as the hardware re-routing of sensor inputs. It allowed using the standard 9144A firmware not only at the lower hierarchical levels but also within the HDC as the 9144A dedicated code has remained totally unmodified.

Host Computer Operation

Any new peripheral design must consider very carefully how the host computer will be able to access its unique features. The 35401A introduces the concept of several independent cartridges which can be accessed one at a time. To ensure that full operational use is made of the device, two distinct operating modes are provided

Sequential Mode

The autochanger will automatically load into the tape deck the lowest-numbered cartridge present in a newly inserted magazine. A local operator can override this selection and force any other cartridge to be loaded by using the front panel controls. Once the cartridge is loaded in the tape deck, the host can transfer data to and from it in exactly the same manner as for a 9144A drive.

Upon completion of data transfers to and from this cartridge, the host computer driver issues a CS-80 UNLOAD command to the drive and (usually) a console message requesting the operator to extract the cartridge from the drive and insert the next one. In

this mode of operation, the 35401A unloads the current cartridge back to the magazine and automatically loads the next one. This allows the host computer to continue data transfers onto this next cartridge, in exactly the same manner as it would do if an operator had performed the cartridge change. The device will thus allow the host to access all present cartridges sequentially, with knowledge only of the currently loaded one. This operating mode is well suited to the unattended backup operation where system and user files are simply being dumped to tape. The autochanger automates media changes as each cartridge is used and its operation continues until all present cartridges have been used or no more data is being sent/received to the device. The autochanger is plug-compatible with the 9144A operation and no new commands are required.

Selective

In this mode of operation, the host specifies which cartridge is to be loaded next. The autochanger will NOT load cartridges until told to do so by the host. There is no local operator intervention via front panel commands to override a host requested cartridge. Once the cartridge is loaded in the tape deck, the host can transfer data to and from it in exactly the same manner as for a 9144A drive. A new CS-80 "LOAD" command has been implemented in the drive to allow the selection of a specified cartridge. The generic form of the command is

```
<01001011> <Number of parameters> <Parameter 1>...<Parameter n>
```

For the 35401A, the number of parameters is one, and the parameter must be in the range 1 to 8 indicating the desired cartridge. The command is only supported in selective mode. Operation in this mode would allow not only the same applications as sequential mode but some rather more sophisticated ones where cartridge selection is necessary. One such application can be to perform a "structured" backup/storage of user files where each group of users (eg a Department) is assigned one or more specific cartridges, or alternatively, one or more calendar dates are assigned to a set of cartridges within a magazine.

Conclusion

This paper is a very brief and simplified overview of the autochanger device. The peripheral offers many user advantages from a compatibility point of view as well as providing access to a large amount of storage for a modest price increase over the single media drive. The concept is clearly not restricted to 1/4" cartridges but a similarly suitable media is a necessity to allow mechanical handling (eg optical discs). The device opens up new areas of applications where operator intervention can be reduced or eliminated altogether, these would include the obvious ones of system backup, structured data storage, software duplication, etc but would also open up other possibilities where -for example- access time is not critical but large capacity at low cost is required.

HP 3000 SERIES BACKUP SYSTEMS PRESENT AND FUTURE

JACK HUFFMAN
HEWLETT PACKARD
GREELEY TAPE OPERATION
700 71ST AVENUE
GREELEY CO U.S.A. 80634

OBJECTIVE

The objective of this paper is to present a general understanding of the backup function on the HP 3000 Series computers by:

1. Examining each element of the current system and the part it plays in the total backup solution. We will discuss both the positive and the negative features of each element in a fair and objective manner.
2. And by using the limiting features of each element of the system as targets for improvement in future development.

Finally, this paper discusses some ideas and options for future improvement to each of the elemental areas of the backup system.

New ideas, re-thinking of old ideas, and investigation of various options happen daily at HP. There is also a flow of new ideas from the general mass storage industry. These ideas are not intended to describe current or planned projects at Hewlett Packard divisional labs and those that are will not be identified in the following pages.

INTRODUCTION

The discourse on each element of the backup system will be at the practical user level and NOT from the laboratory or design level. Also, it is very important to note that the element will only be discussed in its relationship to the BACKUP FUNCTION and NOT to its general computing function.

Let's begin with a quick description of the major elements of the HP 3000 backup system. The major elements are:

1. The operating system and file structure in the area of the store utility.
2. Disc storage.
3. The interface system.
4. Tape drives and alternate types of backup media.

All elements of the backup system EXCEPT the tape drive were designed primarily for user oriented computing, like on-line interaction, file management, number crunching, sorting, and report generation. The backup function is secondary for these elements.

Conversely, the tape drive's primary function is backing up the system. Secondary functions are booting the system, restoring files, reloading and enabling the exchange of data between other computing systems (all off-line functions).

Most elements of the backup system work better when they can transfer data continuously at a constant speed rather than discontinuously, or at faster and slower speeds. A long trip by car is a good analogy. If the trip is taken on the expressway you will go farther, faster, and with less energy if you cruise at a constant speed, not to mention the fact that there will be less wear and tear on the car. This car also does not require high acceleration capability. Contrast that with the car that takes the local route. It races from town to town and through the town the average trip speed is degraded far below the expressway speed. There is more wear and tear on the car during braking and acceleration; a more expensive, higher performance vehicle would be necessary to approach the same average speed as the expressway car.

The above analogy pertains more closely to tape drives since they have the highest mechanical inertia of any element of the backup system. The analogy also pertains to some aspects of disc drive operation like head seeks and finding file extents.

RECOGNIZING LIMITATIONS

Computer systems, and therefore backup systems are, by design, imperfect simply because they represent trade-offs in architecture, cost, speed, and time to market. Furthermore, as soon as a newer technology is implemented the older technology can be looked upon as less perfect. In the following critique of the elements of the backup system we should remember that no system is truly perfect and we can only move forward by recognizing the current limitations and planning to strengthen or eliminate them through new technology.

EXAMINING THE CURRENT HP 3000 BACKUP SYSTEM

MPE FILE SYSTEM AND STORE UTILITY

The first element in the backup system is the HP 3000 MPE file system and Store utility. This is an elegant and flexible system allowing files to overlap many volumes and for file extents to be variable in both size and number. This flexibility is provided transparently to the user and includes highly flexible account management as well as file protection. The limitations to this system in terms of system backup are:

1. All files require time (system overhead) to be opened and closed during a zero dump. This slows down the average data transfer rate from the disc since data does not flow during the time of opening and closing. Perhaps this time could be shared with other activities.
2. File extents are spread over many discs. This is good for a flexible file management system and for on-line system performance, but for backup it causes the system to stop reading from one disc unit and begin with another. This can cause discontinuous data flow and that is undesirable for the backup function.
3. The current MPE Store utility requests data from one disc at a time. Files and extents are read in order and sent to tape in an exact file sequence. This generally limits the data transfer rate of the Store utility to the transfer rate of one channel.
4. The Store utility requires users to leave the system during backup. This is inconvenient for those systems that are required to run 24 hours a day. HP 3000 systems people want the fastest backup possible in order to minimize this delay.
5. Currently there is no provision to switch between two HP7974A/7978 tape drives and rewind one while the other stores data and vice versa.

DISC DRIVES

Disc drives need high transfer rates to enable them to supply data rapidly to the backup system, thereby decreasing backup time. HP has a strong line of current disc products as well as plans for faster, higher capacity, smaller and more cost effective drives for the future.

Disc cacheing has enhanced the overall disc performance by intelligently and selectively cacheing data that will be used repeatedly. The file directory is also cached and this saves many disc accesses during Store.

Rotational position sensing improves data flow during backup by allowing alternate discs to use the channel while another disc is waiting for data to move under its head.

The following items are performance limiters that are continually being challenged. As with all disc drives, faster track seeks and faster spindle speeds would increase access times and transfer rates. Higher linear densities would increase capacity and, at the same spindle speed, increase the burst transfer rate.

HP-IB INTERFACE

The interface transports all of the data to and from the discs and to and from the tape drives. The current band width on the HP-IB interface is roughly 1 MB per second. A faster interface transfer rate would bring a slight improvement to the backup system. In the future, faster peripherals will require a faster interface to balance the backup system.

TAPE DRIVE

The tape drive is the most unusual element of the backup system. It is a serial device in terms of access to data on its medium. It has the highest inertial dynamics and yet must be at rest when data is not flowing to or from it. 1/2-inch reel to reel tape is also constrained the most by world standards largely outside the control of Hewlett-Packard.

1/2" reel tape media are rugged, transportable, archivable, and interchangeable with other computers. It has reasonable capacity and is quite reliable. There are, however, severe limitations on magnetic tape as follows:

1. A major constraint on the tape drive has been its dual role as backup medium and primary medium for exchanging data between different computing systems. The world standard data exchange requirement has put constraints in two primary areas, physical and format.

The physical constraints have made it impractical to design a system for automatically changing reels of tape which would help facilitate unattended backup. Reel changes require an operator.

The world standard format also limits the data density. This density in turn constrains both the data capacity of the medium and the data transfer rate which is a function of tape speed times data density. This indirectly dictates the frequency of medium changes.

2. Like discs, there is a never ending need for increased capacity. This, in the case of tape drives, would diminish the frequency of media changes.
3. Tape media requires a considerable amount of storage space in a well maintained environment.

4. Tape drives have historically been very complex electro-mechanical devices. Sophisticated vacuum column systems or tension arms, with their corresponding servo and control systems, were necessary to synchronize the rapid starting and stopping of the tape (using a capstan motor) with the much slower acceleration of the high inertia tape reels. All of this was necessary to start and stop the media as fast as electronic data could start and stop coming from the computer.

These drives were expensive, needed regular maintenance, and had a high cost of ownership. Streaming drives have eased this problem by providing electronic start/stop capability via intelligent cache buffering. This buffer receives the intermittent data flow from the host while the tape reels are gently ramping up to speed or down to rest.

CURRENT HP 3000 BACKUP SYSTEM.

The backup function is equally important on small, medium, and large HP 3000 systems and it is our goal to offer the best price/performance match in each area.

A consistent input from HP 3000 users has been to make the backup function "transparent" to the users as well as "unattended" for the operator. These are worthy goals.

Transparent backup (sometimes called Dynamic backup) is a function that occurs while users are on-line. Users would probably be unaware that the backup is happening. This would require major changes in the MPE operating system. The need for "fast" backup would be diminished.

Unattended backup, after being initiated, would take place without the need for human intervention. On high-end 3000 systems disc capacities of several GB are common place. The use of high capacity backup devices and/or the automatic handling of media (cartridge changers for example) will be required to meet this need.

The HP35401A 1/4-inch Cartridge Tape Drive was recently introduced by HP with an auto-changing magazine that provides for the unattended backup of up to 536 MB of data. This is an excellent solution for small and mid-range systems especially if they are idle overnight.

At the present time, backup using a GCR drive is the fastest solution for high-end systems. Fast backup minimizes the lack of system availability for users on systems that run 24 hours a day.

FUTURE ENHANCEMENTS AND SOLUTIONS.

GENERAL

We are on the cutting edge of a new era in Hewlett-Packard computer technology. The first HP Precision Architecture machines are soon to be delivered to customers, bringing new strength and growth to an extremely successful product line. These new computers, the HP 3000 models 930 and 950, are only the first entries into this broad product program. These systems will not only offer increased power and performance, they will also offer much larger disc storage capacities and will require much better backup systems than we currently have. At the same time, there are thousands of HP 3000 MPE V systems in the world and we will be improving and supporting these machines for many years. It is our goal to continue improving the vital backup function on MPE V and MPE XL machines as technology allows us to do so.

The remaining portion of this paper will discuss some of the ideas for improvement in all elements of the HP 3000 backup solution.

The ideas presented below come from many people from all HP divisions who are involved in creating solutions for HP 3000 backup systems. In many cases, these improvements are the result of multi-divisional team efforts. Many ideas also flow from the mass storage industry. The message to you, as HP 3000 system users, is that we believe the backup function is important and that there are many areas for improvement. We will continue to examine this area from all sides, optimizing here, making small changes there, and designing newer backup devices as technology allows.

These possibilities for future improvements to the HP 3000 backup system will generally fall into two classes, either FASTER, SMARTER, or BOTH.

"Faster" means speeding up all elements (disc, CPU, interface, and back-up device) of the backup system in various ways.

"Smarter" relates to any improvement where raw speed is not the determinant of success but rather getting the job done "transparently".

Transparent backup would require improvements in the MPE V store utility and to be unattended, would require improvements in backup medium storage capacity as well.

MPE V STORE UTILITY

REEL REWIND TIME SHARING

This technique was used successfully in the past with the 7970E tape drives. The technique is simply to store to a second tape drive while the first one is rewinding and vice-versa.

MULTIPLE DEVICE BACKUP

The logic behind this idea is that if one disc can supply information for a store at rate X, then two discs should be able to supply data at rate 2X, and 3 discs at 3X and so on. (In reality some system overhead performance would be lost and there would be a limit on the CPU.) This would also require multiple (concurrent) tape drives to "balance" the "faster" system unless an extremely fast backup device is also invented.

TRANSPARENT BACKUP (DYNAMIC)

This backup system would be transparent to users of the system, allowing them to continue on-line processing as usual, unaware that the backup is happening.

This idea does not necessarily require any new tape drive or peripheral technology to be created, but would require major changes in MPE. A higher capacity storage device would help the system operator towards an unattended type of operation. The method here would be similar to any other store operation except that before each file is stored, the system would log the exact time and condition. As the file is being stored, the system logs each transaction. When the file is stored the transactions are then appended to it. If a restore of that file is necessary, the system has enough information to read back the file and the changes that occurred during backup and recreate the file.

DISC STORAGE

Rotational Position Sensing has increased the average data rate coming from each disc channel by allowing disc "A" to transfer data on the channel while disc "B" is waiting for data to rotate under its head, and vice-versa. This increases the average disc transfer rate on that channel.

The HP COPYCAT program (currently available) provides for faster backup by breaking the process into two stages. The first stage quickly copies from disc to disc, which allows the system to be returned to the users. The second stage dumps the first stage disc off to tape more leisurely while users are back on the system. Although this does not decrease the total backup time, it significantly decreases the time that users cannot use the system.

Disc cacheing has improved data flow between the disc and CPU for general operation by identifying the more active disc file data and keeping it in cache memory.

HP discs are already quite "smart" with cacheing and Rotational Position Sensing. The following goals are somewhat infinite in nature. The goals are really technical barriers that are continually being pushed back by new technology. They are:

1. Faster seek time.
2. Faster spindle speed and/or higher linear density for a faster burst rate.
3. Smaller volume or footprint to save space.
4. Larger, smarter, and faster cache buffering.

INTERFACE TECHNOLOGY

The following ideas, although general in nature, would also help the backup function by making it more flexible as well as allowing faster transfer rates.

CONCURRENT MULTIPLE DEVICE DATA TRANSFER

This was mentioned in the operating system section. I reiterate it here because the solution is also a function of the interface. The objective is to double or triple the data transfer rate during backup by allowing more than one disc to send file data concurrently.

FASTER TRANSFER RATE

HP-IB has long provided excellent functionality and performance for HP systems but, in the future, both SPU's and peripherals will exceed the HP-IB's current performance range.

FUTURE BACKUP DEVICES (TAPE DRIVES AND OTHERS)

GENERAL BACKGROUND

The transfer rate on tape drives is a function of both the tape speed and the density of the format. At the same tape speed a higher density results in a correspondingly higher transfer rate. The higher density also gives a bonus in the form of higher capacity. In contrast, at the same density, a higher tape speed will increase the transfer rate but does not increase the tape capacity. Higher tape speeds are also mechanically difficult to deal with. Higher density seems to be more advantageous if the product is not constrained by a world standard format/density for data exchange. The past progression of higher tape densities supports this fact.

Backup devices of the future may very well forsake their data interchange capability for some impressive gains in very high capacity or very high transfer rates, or both. 1/4-inch cartridges were developed, despite lack of standardization, because of economy and convenience. Exchanging data may be delegated to telecommunications or networking systems, new 1/2-inch cartridges or even left to the current technologies. As alternatives for data exchange grow, the constraints of standards on backup devices will decrease.

1/2-inch tape drive transfer rates have increased by using higher densities in the following progression: 200, 556, 800, 1600, 3200 6250 and, more recently, 22000 bits per inch net data densities. All drive manufacturers have used different tape speeds with these densities to satisfy various performance needs. These tape speeds have ranged from 12.5 inches per second up to 200 ips.

The pressure to develop lower cost, high performance tape drives resulted in the development of the streaming tape drive. With streaming drives tape motion is provided ONLY by the reel drive motors. There is no capstan and consequently no vacuum columns or tension arms to physically buffer tape movement. Streaming drives ramp up to speed more slowly and then flow tape continuously as data is supplied continuously from the computer. Block gaps and file gaps are written "on the fly". The only problem is that computers do not always provide a continuous stream of data to the tape drive. When the computer stops sending data the drive must also stop to avoid large spaces of blank tape. Since the streaming drive ramps down slowly when data stops coming from the CPU, blank tape flows past the head anyway, requiring a time-consuming reversing or repositioning of the high inertia tape reels.

The next improvement in streamers provided large intelligent electronic cache buffers to manage the intermittent data coming from the computer. This allows the computer to continue sending data to the tape drive even though the drive may be in a time-consuming reposition cycle. Then the data are written from the buffer to the tape when the tape is up to the correct forward speed. These intelligently buffered streamers have proven that they perform as fast, or faster, than their much more expensive start/stop predecessors.

Cartridge technology has been applied to every computing device from terminals and desktop computers, to minis and mainframes. Cartridges come in many sizes and shapes and provide convenience in terms of transportability and reliability because the medium is well protected. In addition, cartridges allow for the handling of media to be automated with changers. The 1/4-inch cartridge is the only one that currently has relevance to the HP 3000 backup system.

The 1/4-inch tape cartridge was successfully implemented for the backup function on small HP 3000 systems 6 years ago. The cartridge was also used to boot up and restore the system as well as for exchanging information between HP 3000 systems. It is worth noting that the 1/4-inch cartridge is not a world standard interchange medium despite the development of the so called QIC standard. Yet it has provided a very economical backup solution for small systems.

This year the power of the 1/4-inch cartridge was increased by the addition of an 8 cartridge auto-changing mechanism providing for the unattended backup of up to 504 MB of disc storage on the HP 3000.

1/2-INCH CARTRIDGES

1/2-inch cartridges will be described in three groups, General purpose, IBM 3480, and Helical Scan.

GENERAL 1/2-INCH CARTRIDGES: Several sizes and shapes of 1/2-inch tape cartridges have been marketed in the computer industry providing more convenience and higher capacity than reel to reel technology. 1/2-inch cartridges generally store less than 500 MB of information. Also, because they typically store information on one or two tracks at a time in a serpentine fashion, the data transfer rate is typically less than 400 K per second. Currently, the capacity on these cartridges is not high enough to offset the lack of industry compatibility and the slow transfer rate.

THE IBM 3480: This drive uses a small 5" by 4" cartridge that will store roughly 200 Mb of data at an average NET data density of roughly 22000 bits per inch. The drive writes 18 parallel tracks that allow data transfers in the 3 Mb per second range. The drive is 4 to 5 times as expensive as a low cost GCR streamer and is, cost wise, more suitable for backup on mainframe and super computers. This "fast" solution would currently not help the HP 3000 backup system because other elements of the system cannot currently support this speed. GCR streaming drives are currently an excellent, low cost match for the HP 3000 backup system.

It is generally believed that the IBM 3480 will evolve into the next world standard format for data exchange. This may not be as important now as it has been in the past because other technologies like data communications and networking are streamlining and simplifying the transfer of information between computing systems with higher speeds and greater reliability. Fiber optics and other technologies will also continue to improve this function. Auto-changers will increase the usefulness of this drive by providing increased unattended backup capability. This technology and many others will be watched carefully for their future feasibility in both cost, and function for HP 3000 systems.

HELICAL SCAN

This paper, for simplicity, will be limited to the VHS video cartridge. The VHS cartridge uses 1/2" wide tape and helical scan technology. Data are written to the tape in parallel lines that go diagonally across the 1/2" media width. Helical Scan has the highest areal density of any type of storage device. It has the capability of being one of the highest capacity backup devices for the future (several Gigabytes).

Current VHS mechanisms have playing times of 2 hours per tape which provide a barrier to raw transfer rate of a few hundredKB/sec.

This drive would be very useful in a "smart" transparent and unattended backup system. Transfer rate is not a problem in a "smart" system. The medium is reasonably priced and readily available. This drive holds some promise as a good economical match for the HP 3000 system in the future.

Data compression, described soon, could help remove the low transfer rate problem of the VHS (helical scan) device and create a very high capacity, "fast" backup solution for the HP 3000.

OPTICAL DISCS

Optical discs are currently available in the market that store up to 4 Gb of data. The only problem is that the media are currently not erasable, and they are quite expensive. Every time you fill up a disc for backup it can never be written to again. The MPE file system would have to be changed radically to be able to deal with write-once devices. Another limiter is that the transfer rate is currently limited to roughly 300 KB per second. This would be too slow for the HP 3000 unless it were used in a "smart", transparent, and probably unattended, backup solution.

Erasable optical discs are not expected for several more years. When they become available they will also have a higher transfer rate than non-erasable drives which is desirable for "fast" backup. Imagine having 4 GB of "fast" storage for unattended backup. A "smart", transparent, MPE backup program would make this a very powerful solution.

DATA COMPRESSION

Several companies in the industry have introduced products that compress data in a variety of ways. The result of this compression is that the capacity of each medium is effectively increased. An additional bonus, depending on the implementation, is that the transfer rate can also be increased. This technique could improve the transfer rates of video cartridges, the optical disc, GCR, or any other backup technology, and enhance their performance for use in either the "fast" backup solution or the "smart" solution.

FUTURE SOLUTIONS

There are many combinations of options for future backup devices. The correct options depend upon other elements of the backup system. Two general solution areas will be discussed:

1. Future options that are possible with transparent backup capability in MPE.
2. Future options that are possible without transparent backup capability in MPE.

FUTURE BACKUP OPTIONS (WITH TRANSPARENT BACKUP CAPABILITY)

Transparent backup is a "smart" solution that would require significant changes in the Store utility of the MPE operating system.

This solution would diminish the need for higher speed in the backup system. The need for higher speed would yield to the need for higher capacity to facilitate unattended backup.

HP 3000 SERIES BACKUP SYSTEMS PRESENT AND FUTURE

HIGH CAPACITY FOR UNATTENDED BACKUP: A consumer video (helical scan) device would be ideal for this solution. Speed would be no problem. Data exchange, however, must be provided for in other ways.

An optical disc will work if it is erasable, or the cost of the medium is decreased. Data exchange must be provided for in other ways.

IBM 3480 technology is currently too expensive for most HP 3000 configurations. The speed is not currently necessary and it would require an auto changer to attain higher capacity

A GCR drive would not have high enough capacity. Data compression would help somewhat, but would not compare with the capacity of helical scan, or optical disc technologies.

FUTURE BACKUP OPTIONS (NO TRANSPARENT BACKUP CAPABILITY)

With no transparent (dynamic) backup users are required to leave the system while the backup is taking place. "Fast" backup is essential to minimize the time that users are off the system during the backup.

If the backup is fast the requirement to be unattended is diminished.

For fast backup to occur, all elements of the system must have roughly the same transfer rate. The following possibilities assume that the disc, operating system, and interface are as fast as the backup device.

For mid-range systems that are currently using a 1600 cpi tape drive the best solution for increasing speed is to change to a GCR tape drive. A GCR drive can increase the transfer rate by roughly 300%.

FASTER TAPE SPEED: The 7978B 75 ips GCR tape drive is currently an excellent match for the current Store transfer rate. In the future, when other elements of the backup system increase in performance, a higher tape speed can be used.

DATA COMPRESSION: Data compression would increase the effective storage density of the current GCR drives and effectively increase the transfer rate. This would be an excellent solution because it would also retain the alternate standard GCR density for data interchange.

HIGHER TAPE SPEED AND DATA COMPRESSION: These techniques combined, could easily double or triple the current GCR transfer rate.

CONCURRENT MULTIPLE BACKUP DEVICES: This was mentioned in the disc section and it pertains to backup devices as well. Storing concurrently to more than one backup device can multiply the total Store transfer rate. This would require changes in the MPE operating system.

HELICAL SCAN WITH DATA COMPRESSION. With data compression this technology could have a faster transfer rate than the current GCR solution. Data interchange capability would be sacrificed, however, unattended capacity would be increased to several GB.

Another possibility would be the IBM 3480 technology. The price would have to be decreased by roughly 70% to be a viable solution for the HP 3000.

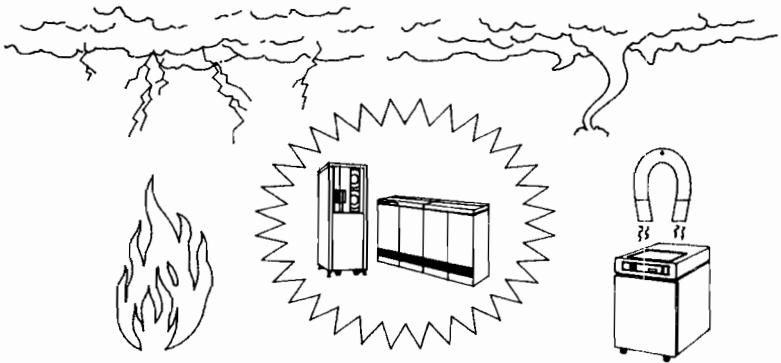
SUMMARY

As you can see, it is recognized that system backup is a vital function for HP 3000 systems. Also, the total solution is a multi-faceted one. The current system is balanced quite well with each element carrying its expected load.

There is a wide variety of options available in both the "fast" and "smart" areas. HP is investigating many combinations of these.

We certainly do not have all of the answers to system backup but we are continually working to improve it. If you have comments or suggestions please tell your local systems engineer or HP sales rep to pass the information along. We'll be glad to hear from you!

Insuring the Future of Your Data by Contingency Planning



Leslie Anne Virgilio
Advisory Systems Engineer
Computer Task Group

Thomas J. Kaminski
Director, Data Processing
SINGER Education Division

CONTENTS

| | | |
|-------------|---|----|
| Section 1. | Introduction..... | 1 |
| Section 2. | System Backup Methods..... | 2 |
| Section 3. | Media Storage - On Site, Off Site..... | 5 |
| Section 4. | Critical Application Identification..... | 7 |
| Section 5. | Recovery Systems..... | 10 |
| Section 6. | Hardware Replacement..... | 12 |
| Section 7. | Disaster Recovery Procedures..... | 13 |
| Section 8. | Rebuilding Your System on Another Computer..... | 15 |
| Section 9. | Licensed Software Concerns..... | 17 |
| Section 10. | Supplies and Auxilliary Equipment..... | 18 |
| Section 11. | Concluding Thoughts..... | 20 |

Section 1. Introduction.

According to Webster's New Students Dictionary, disaster "is an unforeseen, ruinous, and often sudden misfortune that happens either through lack of foresight or through some hostile external agency". To a data processing professional, disaster means the loss of data from a disc-head crash, utility failure, faulty air conditioning, fire, flood, earthquake, hurricane, thunderstorm, tornado, vandalism, sabotage or other occurrences. Most data processing professionals will never see a disaster in their careers and many of them are relying on pure luck to insure the safety of their centers. Any DP center is vulnerable and it will be the unprepared that will panic if disaster strikes. An organized, detailed plan is the key to a successful recovery.

Contingency planning is preparing for disaster (of any kind). A contingency plan should include establishing alternative processing facilities, procedures for restoring critical processing applications on the alternate hardware and establishment of operational procedures, user procedures, and data communications to allow for uninterrupted operation while the original or future site is prepared. A carefully laid contingency plan can significantly improve the ability of a business to survive outages and greatly reduce the length of the outage as well as the cost of recovery.

This paper will not dwell on why a contingency plan is needed. Rather, it will give you enough information to develop your own contingency plan.

Section 2. System Backup Methods.HP-3000 System Backup

The main method of system backup on the HP-3000 computer is through the MPE commands :SYSDUMP, :FULLBACKUP, and :PARTBACKUP. However, there are other methods including:

- ~ Copycat - The HP utility to backup files using removable disc packs
- ~ Backpack - A utility from Tymlabs that is a high speed replacement for the MPE SYSDUMP commands

Whichever you use is up to the needs of the individual organization. The application is the same.

There are two types of backups that can be performed:

- ~ Complete backup - All files are stored
- ~ Relative backup - Only files that have been updated since the last complete backup are stored

The MPE commands :FULLBACKUP and :PARTBACKUP can be used for performing a complete backup and relative backup, respectively. The :SYSDUMP command can be used for both and is sometimes preferred. In the :SYSDUMP command, the systems manager can specify a dump list showing what order the files can be put on your backup media. The default for the :FULLBACKUP and :PARTBACKUP commands is:

```
@.@@
```

It may be desirable to have certain file groups on the front of the backup media by using a list like this:

```
@.@.SYS,@.@@
```

This list puts the files in the SYS account at the front of the backup media for quick access in case information must be reloaded.

The schedule you set up for your organization depends on how vital the information on your system is, how often it is updated, and how much time there is to perform system backups. Generally, you should backup up your files on a daily basis. Figure 2-1 shows a typical backup schedule. Note that:

- ~ On Monday through Thursday, a relative dump is performed backing up files that have been updated from the last complete backup
- ~ On Friday, complete system backups are performed. Since this is the most important backup, two are done. Without a good complete backup, the relative backups following it are useless in most

cases. One system backup is stored off site for further protection.

- ~ Backups are performed at the end of the user's day to reduce the exposure to a loss of data overnight.

Before performing a backup, excess files should be cleared off the system. This includes system log files and editor work files (K-files). To find these files, enter the following file equations:

```
:LISTF LOG###.PUB.SYS,2
:LISTF K#####.@.,2
```

This will cut down on the amount of space needed to store your files and the amount of time it takes the backup to complete.

All jobs and sessions should be logged off before the start of the backup. During a partial backup, some users can be logged on in certain instances, but they may find that they cannot keep an editor file, run certain programs, or may have very slow response time.

After the backup is complete, check the dump listing. Normally, some files will not be backed up and a message such as this will appear:

```
NOT STORED: FILE IN USE FOR WRITING
LOADLIST.PUB.SYS
LOG____.PUB.SYS
MEMLOG.PUB.SYS
SL.PUB.SYS
```

This is normal, these files are being used by the system. Other files may indicate that a session or job was logged on during the backup. If you do not use the standard file list (@.@.@), your statistics will be off at the end (number of files stored, not stored). This is because files that are redundant in the list will be counted as not stored and stored.

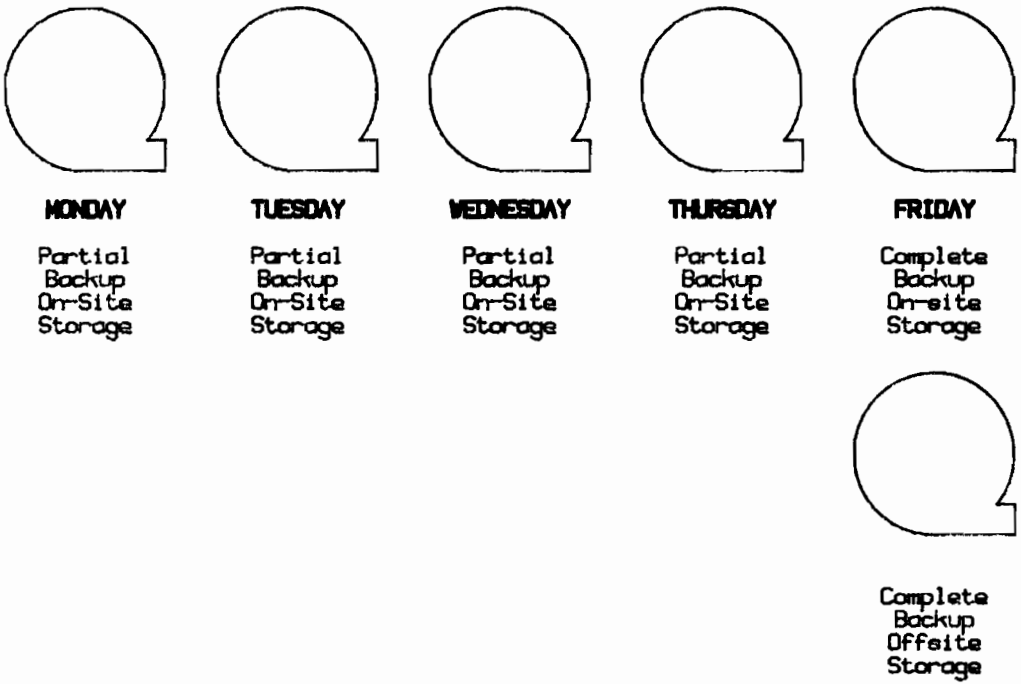
Rotate your backup tapes by using a number of cycles. Four or five cycles are usually good and gives you a month worth of data. Replace your tapes occasionally - ask your tape manufacturer how often.

PC Backup

A primary form of PC backup is using the backup & restore commands to save your data on floppy disk or tape. However, if you have ample room on your HP-3000 disc drives, there is an alternative.

Walker, Richer, and Quinn's series of microcomputer programs (list PC2622) for interfacing to the HP-3000 allow you to transfer groups of files from your microcomputer to the HP-3000. The new reflection software series (Reflection 1, Reflection 3) have a new "plus" option available for using your HP-3000 disc as backup for your microcomputers.

Figure 2-1 Typical Backup Schedule



Section 3. Media Storage - On Site, Off Site

Every organization should have good on site and off site storage of materials and information that would be needed in case of a disaster. These include:

- ~ Your contingency plan
- ~ System backup media (tapes, disc packs, etc.)
- ~ Corresponding system backup listings - useful for finding particular files or systems that you would like to selectively load when recovering from a disaster
- ~ User and System Documentation - instructions for operation of the computer and software
- ~ Software Installation Guides - for any software that would have to be reinstalled on another system
- ~ Special Forms - stock paper can be obtained fairly quickly from local vendors, but special forms may take a long time to obtain

A storage rotation plan should be devised and implemented to insure that material in the storage area is rotated on a regular basis. For tapes, it is a good idea to "cycle" your system backups. For example, you can have four sets of tapes - each week, use a different set (1, 2, 3, 4, 1, 2, 3, etc.). This allows you to always have a set of backup tapes in your storage area, even when current backups are being transported and allows you to reload files that you purged some weeks ago and just noticed missing.

Forms should also be rotated. Many forms, especially multi-part forms, can fade or their carbon can dry out making the forms useless. Some vendors will hold back forms for you. This way, the forms are rotated whenever you place an order (the forms held back are shipped and the last set of new forms are held).

On-Site Storage

Storage of backup media and documentation needs to also be done on-site in case something happens to your off-site files and for those emergencies where files need to be reloaded quickly (processing failures, unwanted file purges). Make sure the on-site storage area that you select is quickly available at all times.

Data Safes are desirable and many sizes are available. Some safes are made for storing paper only. These safes generally keep temperatures below 350 degrees F which is below the flash point of paper, but it is above the melting point of most computer media. A data safe will keep temperatures below 150 degrees F. Safes are rated for number of hours that they will last in a fire.

No matter how good your on-site storage facility, you still must have an off-site facility. A major disaster can wipe out an entire building.

Off-Site Storage

Your off-site storage area should be in another building, away from yours. Some possible locations for an offsite storage facility are:

- ~ Another company - you can set up a reciprocal agreement with another company. They will store your off-site media while you store theirs.
- ~ Banks - banks are sometimes set up to store backup media for other companies. Safety deposit boxes can be used.
- ~ Records Retention Facility - Record Retention Facilities are set up to store both backup media and documentation.

Keep in mind that your backup facility has to be trusted, especially if you are storing crucial data on those files.

Records Retention Facilities can store your media and documentation at a very reasonable cost. For example, prices obtained from a typical facility were as follows:

| | | |
|-------------------|----------------|-------------------|
| Storage of Tapes: | .60/tape/month | |
| Accessions: | .60/item | |
| Document Storage: | .30/box/month | (12" x 10" x 15") |
| Accessions: | .60/item | |

For example, your full backup takes eight tapes, and you want to store your tapes off-site weekly using four cycles of tapes. You have two boxes with all of your documentation in them and four boxes of special forms. The boxes would be rotated monthly. Here is what your approximate costs per year would be:

| | |
|---------------------------------------|---------------------------------|
| 8 tapes x 4 cycles = 32 tapes | for 12 months @ .60 = \$ 230.40 |
| 16 accessions per week (8 in, 8 out) | for 52 weeks @ .60 = 499.20 |
| 6 boxes | for 12 months @ .30 = 21.60 |
| 12 accessions per month (6 in, 6 out) | for 12 months @ .60 = 86.40 |
| | ----- |
| Total Yearly Cost | \$ 837.60 |

Check the hours that the retention facility is open. Make sure you can access your information whenever needed.

Section 4. Critical Application Identification

An important part of your Contingency Plan is Critical Application Identification. During a recovery from a disaster, this process serves the following purposes:

- ~ Aids in making decisions as to which applications need to be brought up first. This is not a simple priority decision. Some applications are more important than others on certain days and therefore "Peak Processing Periods" must be specified to help with these decisions.
- ~ Instructs you on loading the applications onto the computer being used for recovery. Also gives you an idea of what resources the system will take up.
- ~ Gives you an indication of special devices needed for certain applications such as printers, tape, special terminals, and other needs. Also, what supplies will be needed.
- ~ Allows you to set priorities as to what order systems need to be brought up.

Peak Processing Periods are designated for each system and/or subsystem. This information indicates how long an application can be unavailable before it is needed again. Since this information varies from day to day, it is more or less represented in calendar form. Special processing periods (end of month, quarter, etc.) are also specified. All this is taken into consideration when making judgements about data recovery.

On the next couple of pages are two forms that could be used for Critical Application Identification and Priorities/Peak Processing Periods.

Figure 4-1 Critical Application Identification

Instructions: Enter the name of the system and check if any special terminal types are needed and other equipment such as tapes, printers, plotters, etc. For each task that can be run separately within the system, specify hardware requirements (disk space, terminals, printers, other equipment), restore file sets (@.@.PAYROLL for example), system software needed (i.e. POWERHOUSE, COBOL), and any special forms that may be required including their location and vendor name for reordering.

System: _____

Block Mode [] YES Graphics [] YES Personal [] YES
Terminals [] NO Terminals [] NO Computer [] NO

Other Equipment: _____

Disk Number
Space of
(Sectors) Terminals Printer Other
Usage Equip.

Task: _____

Restore File Sets: _____

System Software: _____ File Sets: _____

Special Forms: _____

Location: _____ Vendor: _____

Task: _____

Restore File Sets: _____

System Software: _____ File Sets: _____

Special Forms: _____

Location: _____ Vendor: _____

Task: _____

Restore File Sets: _____

System Software: _____ File Sets: _____

Special Forms: _____

Location: _____ Vendor: _____

Figure 4-2 Priorities/Peak Processing Periods

Instructions: List systems in order of priority. If the computer becomes unavailable on a certain day, at what point must an alternate processing site be obtained? For each system and day, enter the number of hours, name of the day or "NEXT WEEK" that you would need to be recovered by in order for deadlines to be met. Under special processing, list special processing schedules such as end of month, quarter, etc.

| System | Mon | Tue | Wed | Thu | Fri | Sat | Sun |
|---------------------|-----|-----|-----|-----|-----|-----|-----|
| Special Processing: | | | | | | | |
| Special Processing: | | | | | | | |
| Special Processing: | | | | | | | |
| Special Processing: | | | | | | | |
| Special Processing: | | | | | | | |
| Special Processing: | | | | | | | |
| Special Processing: | | | | | | | |
| Special Processing: | | | | | | | |
| Special Processing: | | | | | | | |
| Special Processing: | | | | | | | |



Section 5. Recovery Systems.

The following types of disaster recovery systems are available to businesses in the event the existing computer system is no longer accessible.

Private Backup Site

Private backup sites are owned by the business involved. To be of full benefit in the case of a disaster, this site should be in a different location than the original. There are two types of backup sites; "cold" and "hot". A "cold" site is a fully equipped computer facility, without the computer. Only electrical power, air conditioning, and telecommunications equipment exist. When disaster strikes, the computer and required peripherals must be obtained, installed and tested. Although relatively low in cost, the "cold" site has the disadvantage of a lengthy implementation. A "hot" site is a fully equipped computer facility with an identical or very similar computer system to the original, already installed. Obviously, the most desirable system from an operations standpoint, this alternative is extremely expensive. Another drawback to this alternative is the easy justification using the system/facility for other uses. This eliminates the 100% availability for disaster recovery.

Mutual Backup Agreements

A mutual backup agreement can be between two businesses, or between two different computer sites within the same business, with similar system configurations. They agree to back up one another should a disaster occur. The businesses are usually located near each other. To eliminate competition, the companies are usually in different industries. Although there is little or no cost to the agreement, there are some drawbacks. It is possible, due to the close location of the two sites, that a disaster, such as tornado or earthquake, could occur at both sites. Other problems can arise if one company drastically changes the configuration of their system and the other company does not. Agreements of this type also disturbs the normal processing of the company not affected by the disaster since they will literally have to give their system up to the disaster company for a period of time each day.

Finding potential sites for mutual backup agreements can be done through your local users group. Set up a strong agreement and make sure that you communicate often with your backup site. When the agreement is with an internal organization, control over the computer environment is often easier.

"Cold" Backup Site

The cold backup site is similar to the privately owned cold backup site. It is an "empty shell" facility owned and operated by a company in the business of data disaster recovery. Unlike the privately owned cold site, this site is available to many companies which could cause competition for its use.

"Warm" Backup Site

Computer service bureaus may offer a warm backup site. Arrangements are usually made in advance to allow the business the use of the service bureau in the event of a disaster. Service bureaus tend to be expensive, and ignore special computer requirements.

"Hot" Backup Site

The hot backup site is often the most acceptable solution to disaster recovery. Similar to the private hot backup site, it is owned and operated by a company in the business of disaster recovery. Although there can be competition for its use, disaster recovery companies can often compensate by having several hot sites strategically located. UPTIME, based in California, has a mobile standalone unit that can be placed wherever needed.

Whatever recovery system you decide on, be sure all your computer needs, printer needs, phone needs, etc. are all taken care of. Also, make sure testing of your plan, at the minimum of once a year, can be accommodated.

Section 6. Hardware Replacement.

Hewlett-Packard does not have a written policy for the replacement of hardware. HP sources suggest having a standing purchase order with your local Hewlett-Packard office and in the case of a disaster, they would commit to the shipment of the next available unit. Third party vendors might also be helpful with the availability of used computers and peripherals.

Section 7. Disaster Recovery Procedures.

Disaster Recovery Procedures should include all of the tasks that need to be performed when recovering from a disaster and who is responsible for those tasks. Information in this section includes:

- ~ Phone Notification List - Use a pyramid chain calling list. An example of this is: Caller A calls 2 people and each of those 2 people call 4 people.
- ~ Task Assignment List - A list of tasks that need to be performed in conjunction with disaster recovery. This includes: ordering new hardware and software, insurance notification, and ordering new supplies.
- ~ Transportation Plan - Method of transportation of materials and personnel to the recovery site. If transportation includes using automobiles, please be careful not to put any magnetic media near any of the car speaker magnets. When transporting media, try not to expose it to any severe environmental conditions (i.e. cold) as the media will then have to adjust to recovery site conditions before it can then be used.

There are many types of disasters that can occur to a computer center. Here, those types are broken down into three categories:

1. Building Inaccessible - Fire, flood, earthquake, hurricane, tornado, riots, war
2. Computer Area Inaccessible - Vandalism, sabotage, above reasons
3. Computer Inaccessible - processor failure, disk head crash, utility failure, faulty air conditioning

Depending on the category of disaster, you will need to react in a certain way. Here is an example of what a disaster recovery plan might look like:

| Severity | | | Procedure |
|----------|---|---|---|
| 1 | 2 | 3 | |
| X | X | X | 1. Pyramid Chain Calling - Every contingency plan should have a phone list such as the one described above. |
| | X | X | 2. Have everyone meet together at the office to plan the recovery. |
| X | | | 3. If the building is inaccessible, meet at the place designated in the systems plan to plan the recovery. |
| X | X | X | 4. Notify your backup processing site at this time if it looks like it will be needed. |

- X X 5. Obtain latest backup media and documentation from on-site storage area.
- X X 6. Obtain latest backup media and documentation from off-site storage area.
- X X X 7. Hold disaster recovery meeting. Make decisions on which systems to recover first based on information contained in the Contingency Plan and assign tasks as designated.
- X X X 8. Transport materials using the transportation specified in the contingency plan.
- X X X 9. Recover systems.

Section 8. Rebuilding Your System on Another Computer.

There are a lot of factors that need to be considered when rebuilding your system on another computer:

- ~ Devices - The computer that you are using to recover your data must have the devices required for the systems you need to run.
- ~ Operating System - There may be problems if the software you have uses a different operating system than is on the recovery machine (unless you can reload your entire system on an empty machine)
- ~ Machine Models - Smaller or busier models of the computer may not be able to handle the volume of data in the same amount of time as your computer did.
- ~ System Software - Obtaining Agreements - reference Section 10.
- ~ Data Security - You must insure the security of your data. Make sure you clean up all your files after you are done with the recovery system.
- ~ Compilers and System Libraries - If your application requires the existence of compilers or SL routines, make sure they are available on your recovery system.
- ~ Communication Equipment - Modems, multiplexors, and phone lines are also considerations.

Make sure you know what kind of time is available on the system. With some backup agreements, you may need to set up during odd hours ("C" shift for example).

Make sure you let your backup media adjust to the environment before reloading files on the system. You should probably bring the media into the center as soon as it arrives to facilitate this.

There are two ways to reload files on a system that already has data on it (as in the case of a mutual backup agreement):

- ~ Storing off existing files, reloading your files: This takes a lot of time and it is desirable to have fast backup media (such as a removable pack)
- ~ Reloading among accounts: To do this, you must have dissimilar account names.

If you are reloading your files on an existing system, there are some helpful programs contained in the Contributed Library:

- ~ Account Restructuring Jobs - BULDACCT is a program that builds a stream file that will recreate your account structure on another system. A good idea is to have this run daily. In your plan,

document where this file is located (if you put it in PUB.SYS, make sure you put security on it.

~ UDC Recovery Jobs - There are various programs in the contributed library that will rebuild UDC (User Defined Command) files on your system.

Recovering PC Information

To recover PC information from the HP-3000, make sure your PC transfer program is loaded onto PUB.SYS. With programs such as PC2622, Reflection 1, and Reflection 3, files must be transferred back one at a time. With the "Plus" series from Walker, Richer, and Quinn, you can restore all your files from the disc.

Section 9. Licensed Software Concerns.

Purchased software, whether Hewlett-Packard's or third party, creates another concern in the disaster recovery plan. Licensing agreements prohibit use of the purchased software on any computer other than the one originally purchased for. Disaster has claimed the original computer, now what?

Sources at Hewlett-Packard say that in the case of a disaster, licensed software would be allowed to be used on another system. They warned that software is not compatible between different "MITs" of an operating system. Procedures should be worked out with your local office ahead of time. HP sources did say they would help out in getting the correct version of the licensed software on the recovery system chosen. HP offices keep all versions of licensed software in their local offices.

Sources at COGNOS Corporation said they would be willing to allow movement of their software, including POWERHOUSE products, to a recovery system in the case of disaster. They asked to be contacted before the move is made, if possible. If not possible, they should be contacted the next working day. COGNOS' software is dependent on the series of HP-3000. If you plan to use a different series as a recovery system, prior arrangements should be made with COGNOS.

The assumption might be made that other software vendors have similar policies. The safest thing to do would be to contact any software vendors you deal with when you are developing your disaster recovery plan.

Section 10. Supplies and Auxilliary Equipment.

Also important is all of the computer supplies and other equipment needed to run your systems. If your supplies and equipment have been destroyed, you need to order these items. Your Contingency Plan should contain a list of vendors, purchase order numbers, inventory lists, and other information that would facilitate such a replacement.

A suggested vendor identification form can be found in Figure 10-1. This form shows the name, address, and phone number of the vendor; associated purchase order or account numbers; and items provided by that vendor including standard quantities, prices, and delivery times.

Figure 10-1 Vendor Identification Form

Vendor: _____ PO#/Acct# _____
 Address: _____ Descript _____
 _____ PO#/Acct# _____
 _____ Descript _____
 Contact: _____ Phone: _____

| Part No | Description | Standard Quantity | Cost | Delivery Time |
|---------|-------------|----------------------|-------|------------------|
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |
| ----- | ----- | ----- | ----- | ----- |

Section 11. Concluding Thoughts.

Contingency plans should be well thought out. Thoroughness in testing will save much time and reduce panic in the event of a disaster. Periodic updating of your plan will preserve its integrity.

Remember, a contingency plan is your only key to insuring the future of your data.

IS ONLINE BACKUP POSSIBLE OUTSIDE SPECTRUM ?

Joerg Groessler
Joerg Groessler GmbH
Rheinstrasse 24

1000 Berlin 41 West Germany

Overview

Until today users of the HP3000 are requested to stop their daily work whenever a partial or full backup is performed. With the Spectrum program HP has announced an online backup facility which probably will reduce the downtime caused by backup to almost zero. This facility, however, will not be available to the current HP3000 customers. This presentation will explain two basic approaches to an online backup system in MPE.

What is ONLINE BACKUP ?

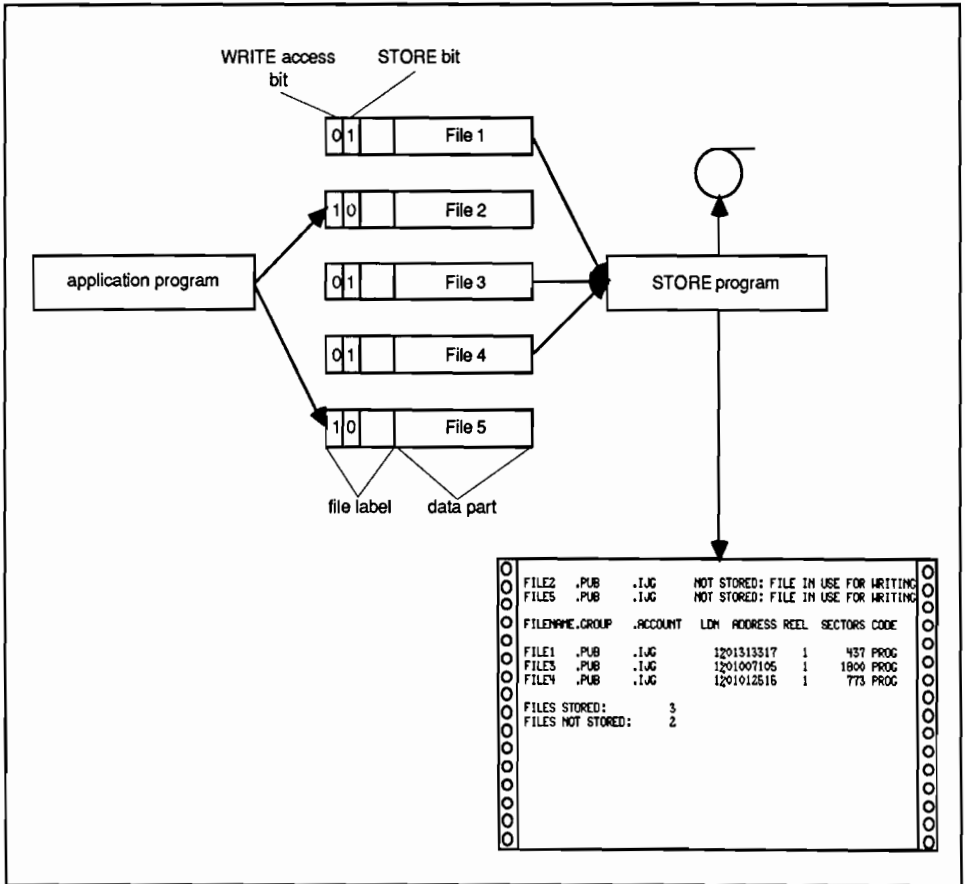
Using the existing STORE facilities (HP's STORE or IJG's BACKUP/3000) the users are required to close files which have been previously opened for write access.

Reason:

- Files cannot be stored in 'zero time'.
- Data which will be written to files will be stored if the file or this part of the file has not yet been stored.
- Some parts of the STORE tape contain more actual data than other parts.

Result:

- Files which are marked as being 'opened for writing' will not be stored.
- A 'STORE bit' is set in the file label to prevent files which are candidates for STORE from being opened for writing.

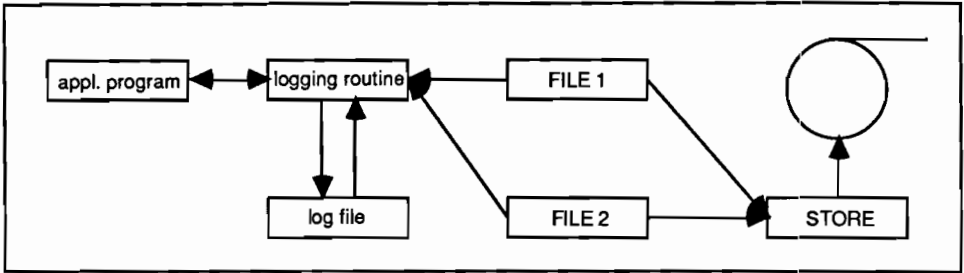


In an ONLINE BACKUP users have to close their files only once at the beginning of the STORE process to put the file system into a defined status. During backup these files can be opened again, even for write access.

To ensure the integrity of the file system the write requests performed during the backup have to be handled differently than usual. This is done with a special logging routine which is called before the actual file write is performed.

First Approach: Actual File I/O Logging

The next picture illustrates the principle of online backup using actual file I/O logging:



- all write requests are performed only into the log file rather than the actual user file.
- read requests by the user program have to check the log file whether this part of the file has already been logged (in that case the data has to be read from the log file rather than the actual user file).
- The STORE program simply stores the original files since they are not changed during backup.
- After the end of the backup the files have to be actualized by the contents of the log file.

Advantages:

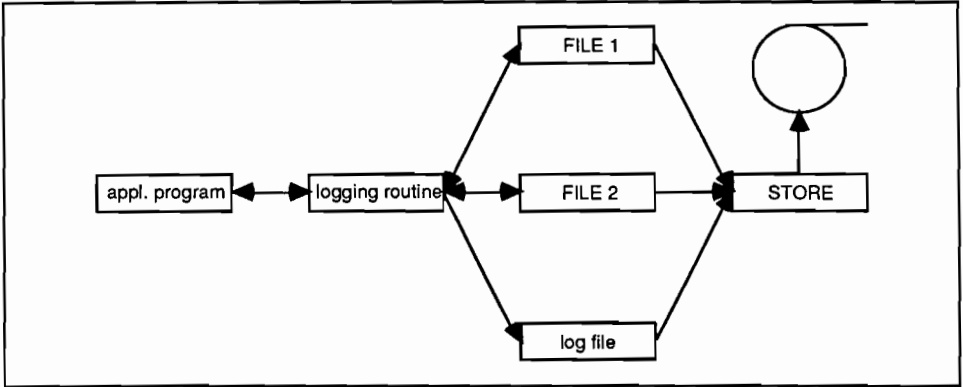
- Minimum overhead for all write requests (one write request to the actual file results in one write request to the log file).
- No changes to the STORE program.

Disadvantages:

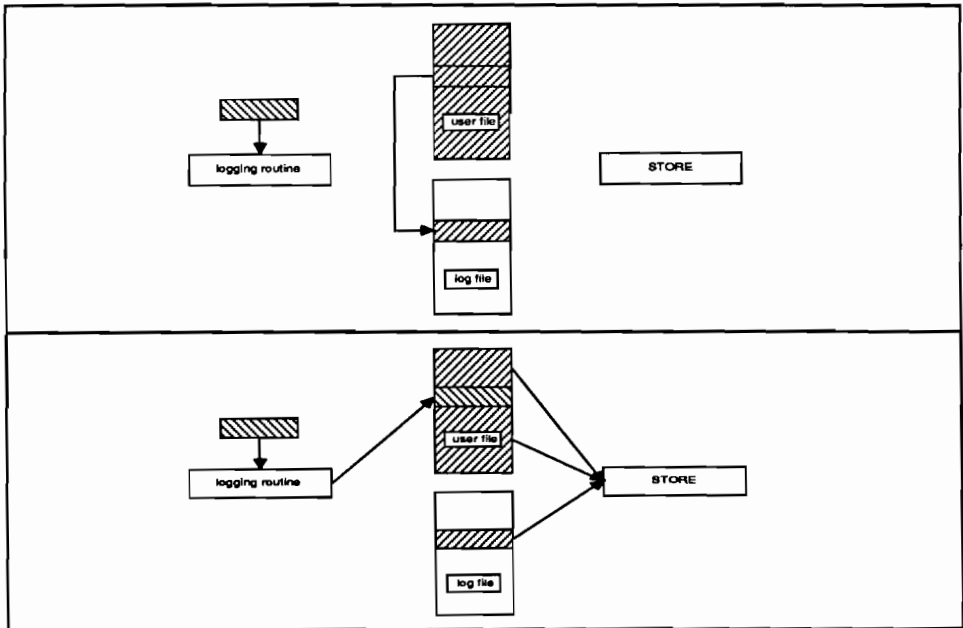
- Overhead for read requests by application programs increases with the number of records already logged and can be very high.
- Hard to recover after a system halt (the files have an old status and the log file might have been destroyed).
- Hard to debug.

Second Approach: Reverse Logging

The next picture shows the principle of 'reverse logging':



- Before a write request is performed the 'old record' is read from the file and written to the log file.
- Each write request has to check if copying the old record is really necessary or if it has been done before.
- The STORE program has to collect its file data out of the actual data file (if this part has not been affected) or out of the log file.



Advantages:

- Much more secure than first approach.
- No overhead for file read requests (they can always be done using the actual file data).
- No additional updating after the end of the backup.

Disadvantages:

- Overhead for write requests increase with the number of records in the log file and can be still high (but not as high as in the first approach).
- STORE program has to check the log file for 'old records'.

Third Approach: ???

Due to the early deadline for this paper there was not enough information available about a third approach at this time. It is more than likely that there is another approach which (if existing) I will explain in the presentation.

Site Wiring: The Foundation for Your Communications Network

*Karen Dudley
Roseville Networks Division
Hewlett Packard Company*

HP provides this paper on site wiring to assist our customers in planning communication networks with high utility and lasting value. Computer usage as a day-to-day tool is rapidly nearing that of the telephone; deregulation of much of the worlds telecommunications companies; a seemingly unlimited variety of communications links and services; digital voice, text and image merging to form a complete communications environment; each of these, and others have combined to transform site wiring into a major strategic issue. The site wiring infrastructure of a buidling or campus is the foundation upon which all site communications systems rest. The problem, simply stated, is that tele and data communications products will come and go, but wire is forever.

THE LAY OF THE LAND

Networking is a complex technology made more difficult by jargon and diversity. Gaining an understanding of the subject sufficient to implement a network, for any purpose, can be a long and involved process. This is unfortunate because in the final analysis, networking is just a tool not a goal. Tools should be a help not a hinderence. To avoid pitfalls, standback and get the lay of the land. Gain a perspective that allows an understanding of your organizations goals, information needs, and current operational characteristics. Internalize this simple concept: a network is nothing more than an electronic way of moving information so that it reaches the right people, in the right place, at the right time.

Building a model for a discussion of networks begins with a look at basic business and environmental factors. *Figure 1* illustrates the primary activities found in businesses today. The activities or tasks that are performed in a given area establish the information needs. Information needs help determine the network technology which can most economically deliver that information. *Figure 2* takes a different view of a business, focusing on the physical environment where these activities are performed. Since networks operate over a physical media (at least with-in buildings) the physical environment will have a significant influence on the choice of appropriate networking technology. Certain environments can also be expected to already have an existing "network" for common communications systems like the telephone. Office environments tend to be clean, quiet and have phones on every desk. Computer centers are custom environments designed for computers at the outset. Factory production areas, by contrast, are electrically noisy, often physically dirty and can be quite large even spreading through several buildings. Unlike the office, production areas have few phones.

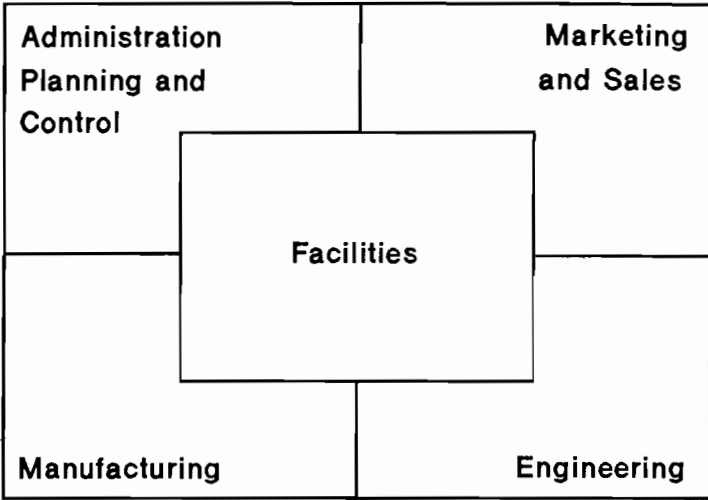


Figure 1: Site Networking Arenas - Activity Focus

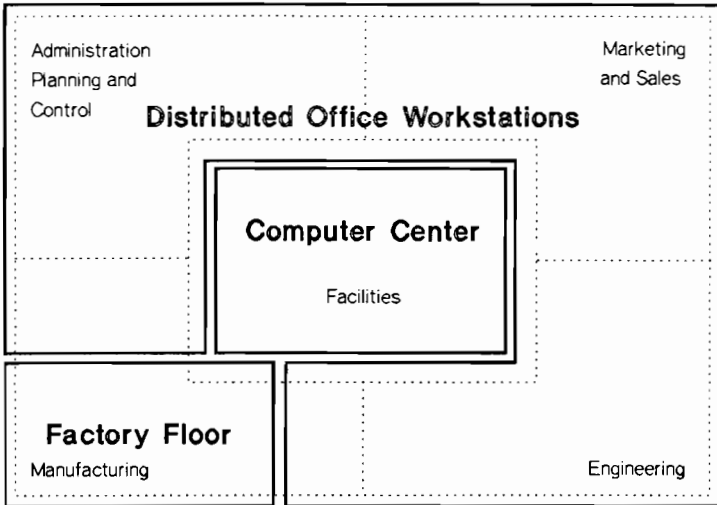


Figure 2: Site Networking Arenas - Environment Focus

Our model is built by determining the unique intersections of activity and environmental requirements which results in the four site networking arenas shown in *figure 3*. HP plans and develops networking solutions for the general office, the engineering office, and the manufacturing plant. In addition, HP addresses the need to link two arenas or blend all three into a complete business enterprise site network. HP recommends several different media options to meet the requirements for all the major site communications systems. Twisted pair, coax cable, and fiber optics each provide effective solutions for selected applications and environments. HP strives to apply each media and communications technology based on the activity, the environment, and a uniform architecture.

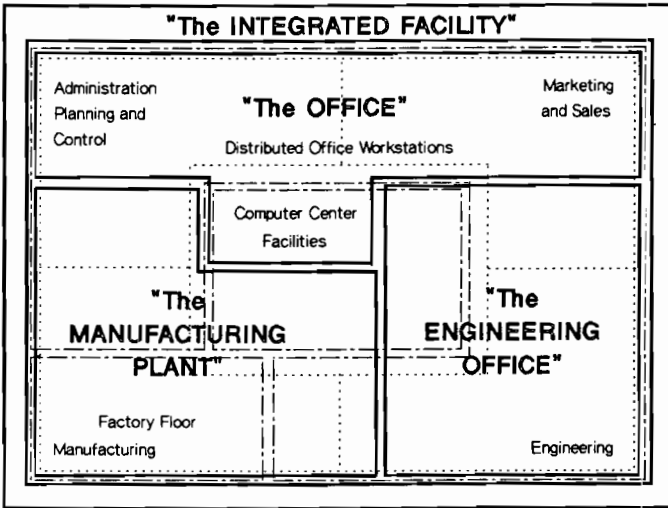


Figure 3: The Four Site Networking Arenas

PLANNING A SOLID FOUNDATION

The OSI Model

There are several important tools that can be a great help in creating a communications network with high, long-term utility. One such tool is the International Standards Organizations (ISO) Open Systems Interconnect Model (OSI) for network architecture. This 7-layer model helps people implement networks according to plan and for standards to be developed at each layer which will ultimately allow equipment from many vendors to communicate easily. However, OSI did not cover everything when it initially appeared. In particular, while it addressed the physical layer which includes media, it did not define an architecture for the media. *Figure 4* refers to this as "Level 0" and it is, in effect, the foundation upon which any well conceived network design must rest. If this foundation is inadequate or missing entirely, it will be impossible to build an effective, flexible and manageable network solution. As the illustration suggests, it would be like building a high rise without first pouring a solid, well engineered foundation.

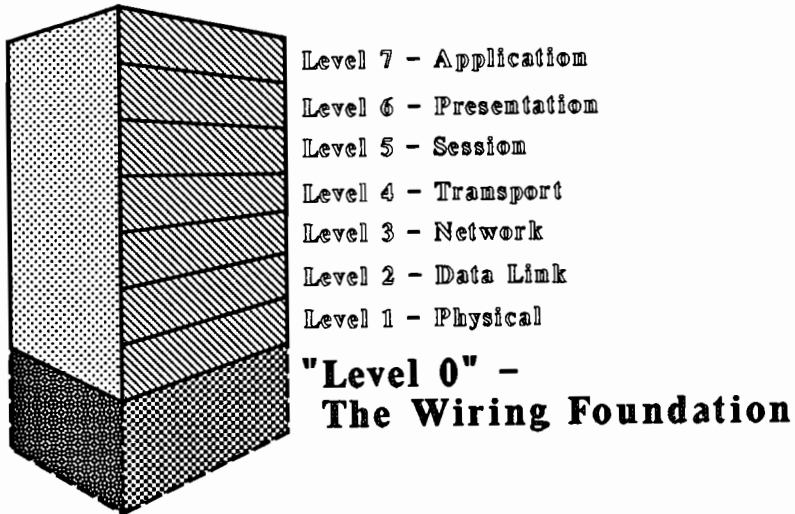


Figure 4: The ISO OSI Tower

A Wiring Foundation Architecture

Figure 5 shows an architecture which addresses the needs of any physical environment and also provides for a way of implementing networks in a controlled step-by-step manner. The architecture defines three elements: the station drop for connecting primarily workstations, terminals and small systems; the hub for terminating and administering station drops; and the backbone facility which provides a physical "conduit" between hubs for appropriate media which will depend on the arena's being addressed. In addition to these three elements, there are several parameters which shape a given implementation of the architecture. There are several types of hubs. Equipment rooms (ER) and data centers (DC) are large hubs designed to accommodate lots of electronics (PBX, head ends, computer systems, etc). They would not normally support any station drops but would connect to the backbone facility. If ER or DC hubs require station drops then the second class of hub would be added called a wiring closet (WC). WC hubs will also be the most common hub distributed through out the facility. They contain passive and active administration hardware and serve as a workarea concentration point for interfacing stations drops to the backbone facility. The third type of hub is the satellite wiring closet (SWC). The SWC is an extension of a WC and gets access to the backbone through the WC to which it is attached. SWC are general employed on large floors to reduce the amount of star wiring required or when remote multiplexers are used to reduce wire needs when space is at a premium.

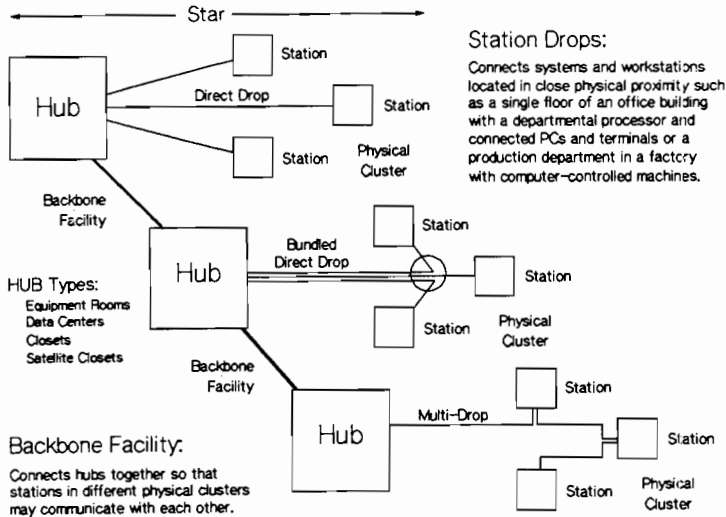


Figure 5: A Distributed Star Site Wiring Architecture

In addition to hub types, another important requirement is the size, number and distribution of hubs. In office buildings, hub will tend to serve a floor (or group of floors) and a large number of stations (100 stations would be considered large). As a result, there will be relatively few hubs in an office implementation. Offices also tend to have a pattern of work locations which is stable (like individual closed offices or a grid of modular partitions) which encourages or requires a permanent installation of station wire. Therefore, station wire in the office needs to be well defined and restricted to a uniform media. Twisted-pair wire is the logical choice since it is obviously needed and often in place for the telephone. The backbone, on-the-other-hand, is usually centrally located, often rising up the middle of a high rise, connecting the hubs. This means changing station wire is difficult and disruptive at best, while changes or additions to the backbone are easier and relatively non-disruptive.

The factory wiring situation, however, is just the opposite of the office. To begin with factories are far more open and flexible spaces. Department and areas are moved more often and the patterns are irregular. To address this situation, more emphasis is placed on the backbone as the fixed element, with more but smaller hubs distributed uniformly through out the plant. Hubs in the factory will literally come and go as departments move. They are, in fact, part of the department they serve and tend to move with it, accessing the backbone at one of the many locations available. This obviously implies that station wire is not permanent and therefore restricting the media used is less important. The backbone, however, is the real infrastructure in the factory network and needs to perform the job of a communications utility. Changes to the factory backbone are very undesirable and expensive.

The last tool derives directly from the wiring architecture. Essentially, the hub serves as a focal point for the creation of departmental or subnetworks. Subnetworks will often operate independently of the facility network, such as a departmental LAN in manufacturing engineering. Sometimes, however, the subnetwork will not function independently, such as a terminal cluster which requires access to a multiuser system in the data center to function. In the figure there are a variety of subnets shown. Each is tuned to the needs and environments of the department or area served. This is one of the major benefits of the backbone and subnet implementation: the right subnet for the job with a versatile backbone for interconnection between subnets. One point worth mentioning, is that some subnets are architectually implemented entirely within a hub. The most common example of this is a data center .

HP is creating an open, multi-vendor wiring foundation that follows the guidelines of an emerging industry standard, EIA TR-41.8. HP's site wiring architecture is a flexible solution that fits the customers application and provides lasting value. HP's commitment to standards ensures that our networking solutions will continue to meet customer needs, today and in the future.

HP AdvanceNet - Multi-Environment Integration
Steve Richardson
Hewlett-Packard Company
19420 Homestead Road
Cupertino, CA 95014

INTRODUCTION

HP AdvanceNet Comprehensive Solutions



Today and Tomorrow

This presentation outlines Hewlett-Packard's solutions and strategy for Information Management through the use of computer and communications networks.

HP AdvanceNet is HP's networking architecture which is based on the OSI model and implemented across HP's computer families. It implements standard protocols and supports communications to IBM. It provides solutions today within the framework of an architecture that will carry you to the 90's.

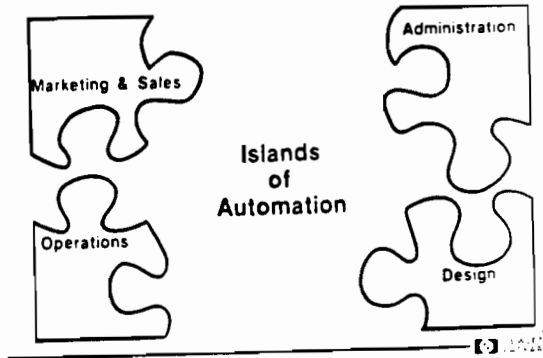
KEY POINTS

HP AdvanceNet is:

- A Networking strategy and architecture.
- Provides comprehensive solutions for information management and communications.
- Solves today's problems and can integrate tomorrow's technologies.

ISLANDS OF AUTOMATION

Network Environment



The drop in price of systems allowed departments to purchase their own systems, and whenever more than one system was used they have built their own network.

With the introduction of personal computers into organizations, the risk of incompatibility and disintegration grew significantly.

When HP introduced its Distributed System Network in 1977 it was intended mainly to solve the single vendor environment. The trend at that time was to computerize the department which might have had several systems that needed to be networked either thru local or long distance links.

In large companies, different departments have solved their problems using different vendors equipment, each having its own systems and networks, thus forming "Islands of Automation".

The illustration shows four typical departments of a manufacturing company as four islands. The upper two "islands" are typically referred as the "office environment", since those applications reside physically in an office. The "Design Applications" island and the "Operations Control Applications" island are other areas that have their own unique requirements and solutions. The "bumps" between the environments represent the interrelationship between the islands. For example, one of the manufacturing applications, Materials Requirements Planning is done by systems that reside in the office environment. In the '80s "integrated systems" is the theme, and one single vendor for all the "islands" is sometimes not very cost effective.

The proprietary networking schemes which met the requirements of the 70's cause a problem in the 80's.

Lack of integration between departments becomes a barrier to further productivity gains. An integrated systems networking strategy consistent with the organization's business strategy is the only way to avoid "Islands of Automation".

A strategy that will help reduce the risk is:

-Focus on a longer term plan for the company systems and networking; choosing a networking architecture that will be able to evolve growth of the company and with the market evolution.

-Definition of codes and standards for the organization.

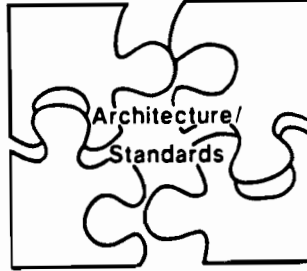
KEY POINTS:

- Departmental systems in the various work environments, created "Islands of Automation".
- There are interrelations between those work environments that call for an architecture for integrated information management.
- Lack of integration is a barrier to further productivity gains.



TOOLS FOR INTEGRATION

Tools for Integration



To facilitate integration between systems families like PCs, HP 1000s, HP 3000s and HP 9000s, and between multiple vendors, there is a need for a networking architecture, and use of standards.

An architecture defines the structure, protocols and functionality of the networks and is the basis for the implementation through products. It provides for integration of functions across an organization.

An architecture by itself cannot provide multi-vendor networking or even multi-system integration. The architecture needs to be based on standards.

Networking architectures are layered. The higher the layer, the more its functions will be oriented toward services to the end user. The standards relate to specific layers.

KEY POINTS

- Tools for integration are: architecture and standards.
- An architecture defines the framework for product development. It defines structure, protocols and functionality.
- An architecture which is implemented through the use of standards can provide multi-vendor networking.

HP ADVANCENET ARCHITECTURE OBJECTIVES

HP AdvanceNet

Architecture Objectives

- Common across HP system families
- Based on the OSI reference model
- Implementation of standard protocols
- Flexibility (multiple protocols)
- Backward compatibility
- Assured growth path
- Communications with IBM and others
- Ease of use

COMMON ACROSS HP SYSTEM FAMILIES

HP AdvanceNet is the first networking architecture common to HP1000, HP3000, HP 9000 and HP personal computers.

OSI REFERENCE MODEL

This seven-layer model is a framework for all of the standards that are being developed, and it is therefore a cornerstone for all of the other objectives.

STANDARDS PROTOCOLS

We will discuss later the importance of standards.

FLEXIBILITY

Today's networks utilize a wide variety of protocols, which will remain the case for the foreseeable future. Implementations based on Xerox XNS, DARPA TCP/IP, Ethernet, and X.25 exist now with newer protocols like 802.3, 802.4, and 802.5 being added. X.400, TOP, MAP and ISDN are envisioned for the future.

Different protocols serve different environments. Some protocols provide comprehensive services while others provide performance. This implies that the ideal networking architecture will need to support various protocols at each layer. One monolithic implementation of a specific set of protocols will not satisfy this need.

As an example, a manufacturing application network might use protocols for 802.4 at the link level. In the same facility, however, a customer might also be using 802.3 with TCP/IP for the engineering workstations and a PBX for connecting terminals to general office computers.

This raises the question of whether an architecture should use one protocol out of many and force the other environments into using it, or whether to allow coexistence of many protocols. The current HP AdvanceNet architecture provides for the use of multiple protocols within any single implementation.

BACKWARD COMPATIBILITY & ASSURED GROWTH PATH

More than 30,000 nodes have been installed using DSN products. When designing the new architecture, HP needed to:

- A. Migrate nodes to the new architecture without a need to rewrite application programs. This was achieved by keeping the DSN user interface in the new architecture, while adding some new services.
- B. Allow DSN nodes that use "old" hardware that cannot run the new software to participate in the network. This was achieved by retaining DSN/DS software as one of the protocols in the new implementation.

The growth path is provided by the OSI compatibility which allows either replacement of one layer without effecting the other layers, and by the support of multiple protocols.

COMMUNICATIONS WITH IBM, DEC & OTHERS

The installed base of IBM systems in many of our customers' corporate offices and the prevalence of DEC in engineering departments required support for communications with these vendors. HP & DEC's commitment to ISO are the basis for providing communications to DEC. Since there are not yet defined protocols for the upper layers, HP has implemented HP AdvanceNet services for DEC VAX Systems. IBM's SNA is not based on ISO and HP had to develop special products to support SNA protocols.

EASE OF USE

Ease of use can be achieved as a function of:

1. Network services
2. Transparent user access
3. Network Management

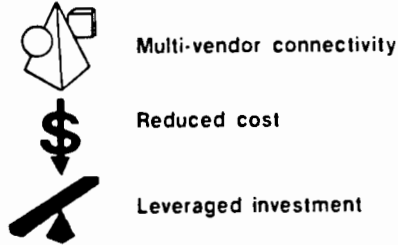
Even when different systems are successfully linked together, networks still require software to do productive work. This software, called "Network Services", provides the tools for data access and transfer across a network.

Transparent access means that to access a network resource, a user needs to know only its name, not a physical location. Network directories provide users with this capability. Directories can reside either on a central system or in a network node. A higher level of service can be achieved when using network dictionaries that describe the data contents of each node.

Network management can contribute to ease of use by managing the network configuration for the users. An end user should not need to reconfigure his or her view of the network whenever the network changes and a new node or new protocol is added.

WHY STANDARDS

Standards



Throughout our discussion we've mentioned the importance of industry standards, as the second key element for integration, complementary to the architecture. With standards, customers are more assured of **MULTI-VENDOR CONNECTIVITY**. This allows them to choose the best equipment from a variety of vendors to serve their application needs.

Standards also significantly **REDUCE THE COST OF CONNECTIVITY** providing for easier communication among various types of equipment (PBX, CPU, workstations). Standards also facilitate cost reductions through VLSI. For example, IEEE 802.3 standard allows manufacturers to purchase VLSI components on the open market.

Third party products, adhering to standards, can complement HP's solution, allowing HP customers to get the best price/performance for a specific problem.

Hardware investments can be reduced because the same transport and network services can be transmitted over different physical links.

Another customer benefit that standards provide is that customers can **LEVERAGE THEIR INVESTMENT** in network software and hardware. Since standards are relatively stable, customers can depend on the upgrade/expansion capabilities of an implemented standards network.

Standards implemented within the OSI model allow replacing or adding a specific layer standard without effecting the applications.

This commitment to standards is, however, no small task on the part of a computer vendor. It implies major challenges in testing and support of multi-vendor networks. HP is fully committed to accept this challenge to work in a multi-vendor environment.

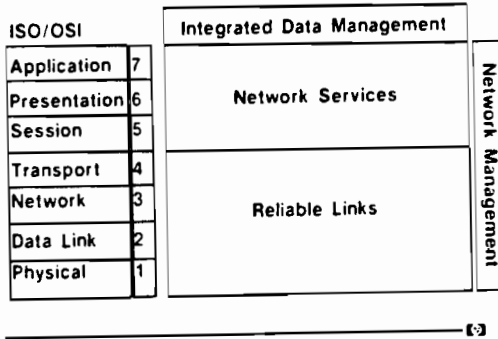
KEY POINTS

- Provides for multi-vendor connectivity.

- **Reduces cost:** VLSI, sharing of links.
- **Leverage investment:** more stable, replaceable layers.

HP ADVANCENET ARCHITECTURE

HP AdvanceNet Architecture



A networking architecture is the basis for integrated information management systems.

HP's networking architecture facilitates integration between different computer families like HP 1000, HP 3000, HP 9000 and PCs, and is the basis for multi-vendor networking and for integration of the various work environments.

The International Standards Organization (ISO), published in 1979, a proposal for a networking architecture, named "Open Systems Interconnect Reference Model" abbreviated as OSI.

The OSI Reference Model is used by standards organizations like CCITT, IEEE, ISO and National Bureau of Standards (NBS) to develop standards for each of the layers. The model is referred to as "Open System" since it is publicly available and provides for communication between different vendors that will implement the same architecture and standards.

This model has seven (7) layers, starting with the "physical layer" which identifies the physical connection (The RS232 Standard serves this layer), and ending with the "application layer". In the illustration we grouped the layers into two bands to simplify the discussion.

- * **LINKS** - These lower layers are the visible layers of the architecture, they are implemented mainly in hardware and have pretty well defined standards as we will see in a later slide.
- * **NETWORK SERVICES** are those software modules that provide useful functionality to the user on the network. The most common network service is File Transfer. This allows a file to be transferred between heterogeneous computers, an HP 1000 to HP 9000 for example, without having to worry about the link technology used to connect those systems. Virtual terminal is another useful function, allowing any terminal on the network to access any system on the network. Remote Data Base, File Access and Remote Process Management are other network services provided by HP. One of HP AdvanceNet

contributions was defining common network services across HP product lines. The common network service available TODAY across all the HP computer families is Network File Transfer.

- * **INTEGRATED DATA MANAGEMENT** - To be able to easily use services in a networking environment, transparent access to data and management of data and resources have to be provided.
- * **NETWORK MANAGEMENT** - Network management is perceived generally as managing the link layer only, and indeed the links are the main contributing factor to problems in a network. In the future, multi-vendor networks and multiple protocols will require a network management architecture that will provide tools for problem determination in the various layers. HP AdvanceNet includes a network management architecture that provides this capability.

KEY POINTS

- HP AdvanceNet architecture is based on the OSI Reference Model, which is a 7-layer model.
- The architecture defines common network services for HP systems (HP 1000/3000/9000 & PCs) to facilitate the integration of the various work environments.
- The architecture adds to the OSI model a network management architecture and an integrated data management services.

LINK STANDARDS

Reliable Link Standards

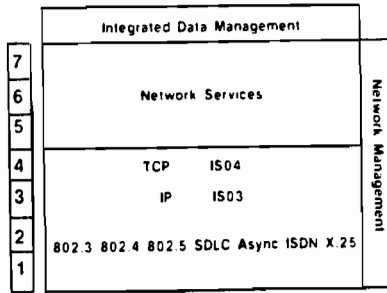


FIGURE 1

Today, certain standards are established or are emerging for communication networks. At the link level, standards have been well defined. However, at the higher ISO levels (i.e., services) standards are in the process of being defined by various standard committees.

Since link standards are more easily defined than software standards, software standards are the limiting factors in the goal of full multi-vendor compatibility.

PROCESSOR-TO-PROCESSOR & WORKSTATION-TO-PROCESSOR LINKS have standards such as 802.3 for LAN's and X.25 for wide-area networks. IEEE 802.4 received acceptance as part of the GM MAP model and IEEE 802.5 was endorsed by IBM.

Presently, each **TERMINAL-TO-PBX** communication system is unique and is using various asynchronous protocols, but may evolve to a standard that is based on ISDN (Integrated Services Digital Network). ISDN is a standard being developed in Europe for integrated voice and data networks.

A standards committee is working on a **PROCESSOR-TO-PBX** standard, which, if adopted, may potentially be a similar to DMI (Digital Multiplexed Interface), a standard based on the developing ISDN standard that is supported by several computer and PBX vendors.

Layers 3-4 provide routing and data integrity and flow control. TCP/IP are protocols which were used by the US Department of Defense ARPA network and became an industry standard implemented by various vendors including DEC and HP. Recently, ISO has developed standards for these layers and HP has committed to implementing these standards as part of our commitment to TOP and MAP.

KEY POINTS

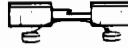
- Link standards are well defined:
- For Wide area Networks: X.25, BSC, SDLC

- For Local Area Networks: IEEE 802. 3, IEEE 802. 4, IEEE 802. 5
- For Voice and Data: ISDN is in the process of definition.
- Network and Transport Standards (layers 3-4) have been recently defined by ISO.

NETWORK SERVICES

Network Services

- Network file transfer



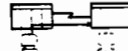
- Remote data base access

- Remote file access



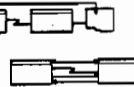
- Virtual terminal

- Remote peripheral access



- Network interprocess communications

- Remote process management



© 1986

Network Services are the services that allow end-users and applications to perform productive work through the network; i.e. data transfer and data access. Since no standards were available when HP AdvanceNet was implemented we used the DS proprietary network services with several enhancements.

These services include:

NFT is the common network service available in 1986 across these systems/ workstations families.

NFT - Network File Transfer - Provides for moving of files from one node to another.

RDBA - Remote Data Base Access & RFA -Remote File Access - provide the ability to access remote files and databases as if they were local.

VT - Virtual Terminal - Makes a remote terminal to appear as if it attached locally.

Remote Peripheral Access - The ability to access peripherals attached to a remote system as if they were local.

NIPC - Network Interprocess Communications - Allows processes in different nodes to communicate.

RPM - Remote Process Management - Allows to run, monitor and terminate a process from a remote process.

LLA - Level 2 Link Access - Allows to access programatically the Level 2 link layer.

There are additional family unique services like:

- Remote Command Processing on the HP 1000.

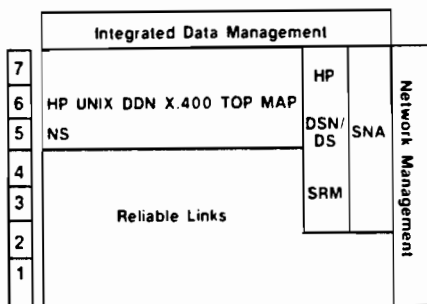
- UUCP on HP 9000.

The services on HP 3000 include: NFT, VT, RDBA, RFA, RPM & NIPC. On HP PCs: VT, NFT, IPC & MS-RFA. On HP 1000: NFT, NIPC and LLA. On HP 9000 (200/300): NFT, LLA and RFA under HP-UX. HP 9000/500 has additionally also: NIPC & RPM.

NFT is available TODAY, providing a common network services across all the HP system families.

Network Services Protocols

Network Services



Protocols for Network Services are evolving and not too many standards exist today. It seems that this will also be the case in the near future, so the question to ask is how we can protect a customer's investment in today's protocols, if there are good chances that new protocols will evolve later on. The HP AdvanceNet solution is an architecture which allows coexistence of multiple protocols within each layer. The illustration presents examples of such protocols.

HP NS - are the HP proprietary Network Services.

UNIX - are UNIX network services.

DDN - are Department of Defense Network services.

X.400 - are CCITT Standards for messaging.

TOP - are the Technical Office Protocol services.

MAP - are Manufacturing Application Protocol services.

Although HP does not have products today that support the last four services, we are investigating supporting them in the future.

HP DSN/DS - is the former proprietary architecture used for HP systems networking. The new architecture allows coexistence in the same network of "old" DSN/DS nodes and new HP AdvanceNet nodes. Rather than forcing old nodes to upgrade to the new architecture, we allow them to be part of the network. "New" nodes that need to talk to "old" nodes need to have the old protocols installed as part of the new system.

SRM - Is the proprietary Shared Resource Manager providing sharing of peripherals for HP 9000 BASIC/PASCAL work stations. An HP 9000 UNIX server can link to an SRM, thus allowing BASIC/PASCAL nodes to communicate to UNIX workstations.

SNA - SNA is an example of another architecture which was developed by IBM prior to the publication of OSI. SNA is a proprietary architecture and it fits mainly IBM mainframes. HP supports connectivity between HP 3000 and SNA mainframes.

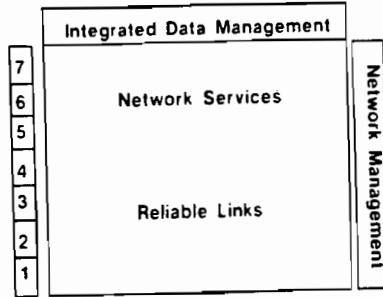
SNA is not an architecture for multi-vendor connectivity since it uses IBM proprietary protocols in the various layers, and it is designed according to IBM's definition of Distributed Data Processing and networking. For example, HP systems can not communicate to other HP systems or to DEC using SNA. DIA/DCA was also added to the SNA architecture as new proprietary protocols. HP has announced plans to support these functions as part of the commitment to support communications to IBM

KEY POINTS

- HP AdvanceNet allows coexistence of multiple protocols. This provides benefits such as: Participation of "old" HP/DSN/DS nodes in the network, easy migration to new protocols and catering to various needs.
- HP AdvanceNet provides network services within and between HP system families and to other vendors such as DEC and IBM.
- Compatibility with HP/DSN/DS network services is preserved, so applications need not be changed when nodes migrate to the new architecture.
- HP supports communication to IBM on HP 1000/3000/9000 and PCs. Communication to SNA is available on HP/3000s for for both batch (SNA/NRJE) and interactive (SNA/IMF) communication.

TRANSPARENT INFORMATION & RESOURCES ACCESS

Network Management



A vital ingredient of a successful network is transparent access to information and resources within the network and management of the network. Although this is not part of the OSI model, HP added these functions to the HP AdvanceNet architecture.

TRANSPARENT INFORMATION AND RESOURCES ACCESS

Transparent access increases the ease of use of resources and information in the network. Transparent access means that information or resources are available to the user without knowledge of their location, without knowledge of information's structure or storage, and without knowledge of the topology of the network.

Several tools are necessary to provide transparent access, including network directories, data dictionaries, messaging services, and applications built on top of these tools.

A lot of evolution is taking place in the area of network directories. Directories currently exist for HP 3000 and HP 1000 based networks. These directories are primarily used by HP networking products to locate paths to desired nodes. Since directories are vital for navigating through networks, the standards organizations are turning their attention towards directories, as well as protocols and services. ISO, MAP, TOP and others are all discussing directories, but firm standards have not evolved yet. As these standards emerge, HP will assess what is necessary to provide industry standard network directory services.

While network directories describe the topology of a network and contain some information about the nodes in the network, it is the data dictionary's role to maintain information about the data in the network. A data dictionary contains "metadata", or "data about data". This includes such information as the location of data (what node), how the data is stored (i.e., in a flat file, a data base, ...), how the data is accessed (what is the key or path to the data), and the format of the data (type and length).

Programs such as ad-hoc inquiry tools, fourth generation languages and user written applications make use of the data dictionary by allowing the user to request data items by

name, and then relying on the data dictionary to provide the "where" and "how" for accessing the data.

Currently, HP has two data dictionaries, Dictionary/V (formerly Dictionary/3000), and System Dictionary (/V and /XL). Dictionary/V has been available for several years and is an integral part of the RAPID family. The recently released System Dictionary is HP's next generation data dictionary, providing greater functionality and flexibility. Both dictionaries are based on the "entity - relationship" model, where entities are objects (data items, files, programs, etc.) and the relationships are linkages between entities (such as "file contains data item").

A core set of entities and relationships are supplied with System Dictionary. This core set enables the dictionary to maintain information on programs, nodes, VPLUS screens in addition to information about data and files. Thus, System Dictionary can maintain information on just about anything in the network, and make that information readily available to other tools, applications, data administrators, system analysts, etc. Additionally, the System Dictionary is extensible; that is, if users wish to maintain information on entities or relationships that aren't supplied in the core set, they can simply create the new entities and/or relationships.

Another tool for transparent information and resource access is an asynchronous, queued messaging service. ("Asynchronous" in this context means that the recipient of the message need not be immediately available.) Such a service delivers messages or files between processes. A "sending" process only needs to specify the name of the process which it wishes to send a message to, not anything about which node the receiving process is on, the topology of the network, or the current availability of the network.

This type of messaging service not only provides network transparency for processes and programs, it also manages the queuing and delivery of messages automatically.

NETWORK MANAGEMENT

Network management is part of the HP AdvanceNet Architecture. Network management encompasses five basic areas: network status/diagnostics, network performance, network administration, remote operations, and planning and simulation.

Today's network management products comprise node management software which is part of the network products, and test equipment, for network problem determination. The networking products which comprise Network Link products and Network Services products have built-in management utilities for node management: configurators, tracing and logging, and diagnostic tools.

For private X.25 networks, HP is jointly marketing with M/A-COM, a Network Control System that provides traffic statistics and utilization rates, on line configuration changes and diagnostics tools.

HP is a leader in test equipment for network problem determination. This includes the 465x family of protocol analyzers, BERT (Bit Error Rate Tester) for digital testing and TIMS (Transmission Impairment Measurement Set) for analog testing.



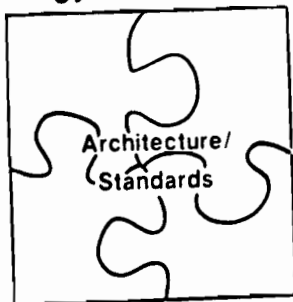
HP is committed to providing additional and more powerful network management capabilities for leased line networks.

KEY POINTS

- Transparent access is built in the Network Services products. It will be enhanced through the System Dictionary and a messaging service.
- Network management capabilities are built into the HP AdvanceNet products to provide node management and diagnostics.
- HP is a leader in test equipment for network problem determination.
- HP will continue to strengthen its offerings in the areas of transparent access and network management.

STRATEGY FOR INTEGRATION

HP AdvanceNet Strategy for Integration



HP AdvanceNet architecture and HP's commitment to standards are HP's strategy for integration. This will serve the need for a networking architecture that will act as a framework and standards which mold contents into the framework and enable multi-vendor networking. The AdvanceNet strategy is to support sophisticated communications networks by distributing the intelligence among the various network elements.

The intent of our network architecture is to provide a focal point for present and future development of data communications products within HP. In its layered approach, it provides a common method or protocol for exchange of data and a wide range of communications solutions. As a first step it increases the integration between various HP product lines, as more standards evolve it will provide true multi-vendor networking.

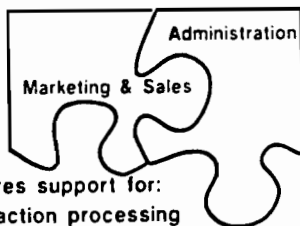
KEY POINTS

- HP AdvanceNet strategy supports sophisticated communications networks.
- The architecture and the commitment to standards provides for integration across the work environments.

THE OFFICE ENVIRONMENT

TRENDS

The Office Environment



Requires support for:

- Transaction processing
- PC integration
- Voice/text/image integration
- Multi-vendor equipment
- Easy & transparent access

In this environment, we refer to those applications which physically reside in an office. These include operations, planning, control, accounting, word processing, sales analysis and other applications unique to your industry. Industry consultants and our customers told us that there are several trends that can be identified in this environment:

- * **Growing ADMIN costs.** To compete successfully, companies need to provide better services at a lower cost. ADMIN costs are the fastest growing item which is not directly related to production, and is the most difficult to control. This increased the awareness to office automation and cost effective technologies. Communications costs such as meetings, travel, and wire services are not declining.
- * **PCs are emerging in every organization.** The sharp decline in cost of PCs and the broad awareness of their existence, make them almost a "status symbol" in the office. They replace stand alone office products and computer terminals. Organizations still struggle with the question of how to increase personal productivity using PCs.
- * **International standards are emerging for links (IEEE 802), networks (X.25), voice/data integrations (ISDN), etc.** Users need to be assured that their investment today will be able to incorporate the evolving standards in the future.

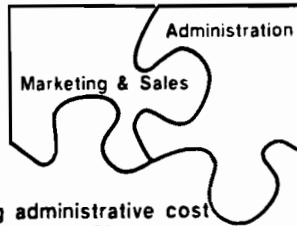
KEY POINTS

- Competitive pressures demand a decrease in administrative costs, but these costs are growing.
- Personal Computers are emerging in every organization, which increases personal productivity.

- International standards are evolving and have to be considered in every networking strategy.

THE OFFICE ENVIRONMENT

The Office Environment



Trends:

- Growing administrative cost
- Emergence of PC's
- Emergence of international standards

* REQUIRES SUPPORT FOR

Increase in Personal productivity is our customers' main requirement. The technology implications are as follows:

* TRANSACTION PROCESSING:

The need for better service and cost control led to transaction processing applications on minicomputers, which provided current data more easily. These involve system-to-system and system-to-workstation communications over both short (Local Area Networks) and long distances (Wide Area Networks).

* PC INTEGRATION

PCs are used for document preparation and printing, spreadsheet applications and graphics. Today, however, users face an array of information access protocols and resources, as well as separate devices for personal computing, office automation and central data processing. The PC, as the personal office tool, needs to be integrated into the departmental systems applications so that personal productivity can leverage departmental resources. In addition, individual PC's are being connected into smaller workgroup systems which are linked to larger departmental systems.

* VOICE/DATA/TEXT/IMAGE INTEGRATION

Providing voice communication is still one of the major expenditures in an office environment. To reduce data communication costs, there is a need for utilization of voice circuits also for data, text, and graphics communications. This is needed both in wide area and local area networks.

To increase productivity two additional features are required:

- Functional integration, i.e. use the workstation for phone management.
- Full voice/data integration, i.e. use digitized information to carry and store information as in "Voice Annotated Mail" and "Voice Store & Forward"

* MULTI-VENDOR EQUIPMENT SUPPORT

The type of equipment that typifies this environment includes mainframes and minicomputers, standalone word and document processing equipment, printers, plotters, PBX's, telex, videotex, and an extremely widespread installation of telephones. Large corporations are composed of many divisions and subsidiaries with various types of systems who need to share and consolidate data. Multi-vendor standards are therefore imperative to allow for different equipment to communicate easily. Mainframe access is needed in most organizations.

* EASY AND TRANSPARENT ACCESS INCREASES PERSONAL PRODUCTIVITY

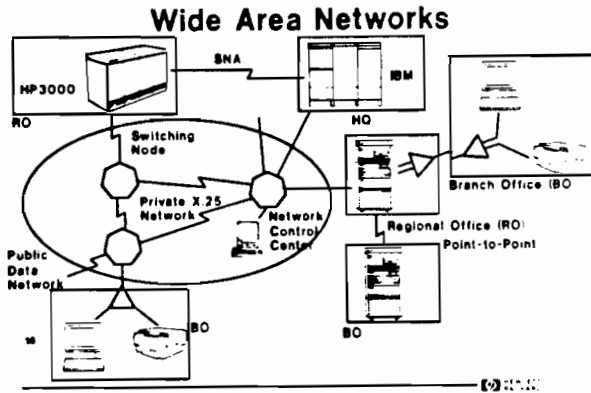
Most managers and their staff are not DP professionals but they need easy access to local and central applications. By transparent access we refer to the capability to access peripherals, data bases, and applications without worrying about their physical location and type of connection.

KEY POINTS

Increase in productivity is the main requirement, and the technical implications are:

- Transaction processing leads to higher quality service.
- PC integration increases personal productivity.
- Voice/data integration reduces communications costs.
- Multi-vendor equipment support reduces cost and increases productivity.

WIDE AREA NETWORKING



Wide Area Networks (WAN) link computer systems and workstations in a multi-site environment, as opposed to Local Area Networks, which are concerned with networking in one site.

HP's Wide Area Network strategy is based on X.25 packet switching networks. X.25 network solutions increase connectivity for computer systems and workstations located anywhere in the organization. X.25, being a true international standard adopted by most computer manufacturers, provides the basis for multi-vendor communications. As a company backbone network, this will transport also IBM SNA traffic between IBM systems and terminals (not shown in the illustration). X.25 networks also allow for a significant reduction of the communication costs, as compared to point-to-point networks. In addition, they provide the reliability and security required by critical applications, extensive control and growth.

HP offers a full X.25 network solution with products that allow HP computers and workstations to access an X.25 network, as well as a powerful private X.25 network transport.

The X.25 link products provide X.25 connection capabilities for the HP 3000 and HP 1000 families of computers. The HP 2334A X.25 multiplexer allows groups of workstations (terminals, PCs, printers, ...) to access remote computer systems. These HP X.25 products have been certified with the major public X.25 networks all over the world (29 networks in 24 countries).

HP also distributes and supports the HP30290A X.25 line concentrator (DYNAPAC Model 8 Multi-Switch) (in the US only). This is typically used to concentrate on one link several HP2334 PADS and HP 3000 systems.

Under a joint marketing program established for HP customers, M/A-COM delivers its technologically powerful CP 9000 Series II private X.25 network, while HP backs the network with its worldwide support and service that is rated number one in the industry.

The M/A-COM network is composed of switching nodes tied together by a set of leased lines and other transmission links and a network control system.

In addition to its comprehensive X.25 family, HP also offers remote point-to-point links for its computers which give optimal performance in limited network configurations with a small number of connected devices. The HP Satellite Link on the HP 3000 is best suited for very high volume long distance connections.

The illustration shows a wide area network of a company which has IBM equipment in its Corporate Headquarters and HP systems in its Regional Offices (RO) and Branch Offices (BO).

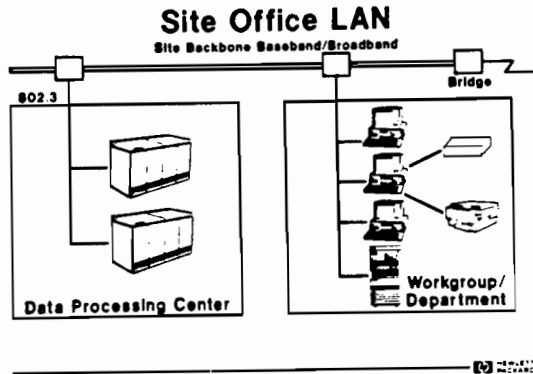
To further increase the reliability and serviceability of the network, HP provides a comprehensive line of test equipment for network monitoring and analysis.

To communicate between HP and IBM systems, HP 3000 has two products: SNA NRJE for batch transfers and SNA IMF for interactive communications.

KEY POINTS

- HP has selected X.25 as the basis for its Wide Area Networking strategy.
- X.25 provides the most cost effective and widest range of alternatives for remote data communications and is the basis for multi-vendor networking.
- HP has a full X.25 product offering including a private X.25 network and the network access products for its computers and workstations.
- HP provides products to communicate to IBM using SNA communications.

SITE OFFICE LAN



Local Area Networks (LAN) emerged in the last years as a cost effective high performance network for a building or a site comprising several buildings.

IEEE 802.3 was the first standard for LANs, building upon the widely used Ethernet protocols.

A site backbone LAN is a cable which runs through a site or a building and provides utility for data (baseband) or data, voice, text and video (broadband). A site backbone can be baseband (i.e. all the cable bandwidth is dedicated to one node at a given time) or broadband (i.e. the bandwidth is split to subchannels).

Out of the facility backbone cable, branch out local networks such as LAN/3000 which is an IEEE 802.3 baseband LAN connecting multiple HP 3000 in a computer center, or OfficeShare that enables PCs to share peripherals and data or communicate to a departmental HP 3000. To connect local branches into a backbone, a Repeater or Bridge is needed. A bridge is used if the backbone and the branch use different protocols. HP offers a repeater today.

HP has certified Ungerman Bass products for connecting workstations and systems to a broadband backbone using RS-232 connections.

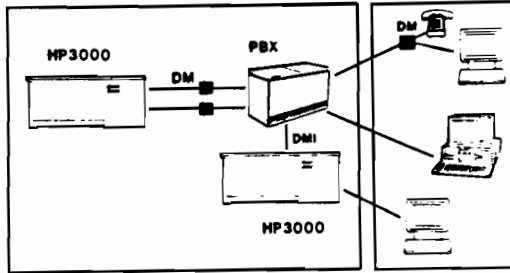
KEY POINTS

- A Local Area Network based on IEEE 802.3 is available as a backbone link, as a computer center local link, as a workgroup (OfficeShare) subnetwork.
- LAN IEEE 802.3 is a standard for high performance communications for workstations and systems.

- Higher layer software provides for data access and resource sharing between PCs and systems.

PBX

PBX Local Solution



In our review of solutions for the office environment, let us start with local solutions: the PBX and the LAN. The emergence of the personal computer coupled with the already widespread use of the telephone in this environment provides the option of integrating PCs into telephone technologies to create an information network. Businesses are taking advantage of existing phone lines to create local communication networks.

The PBX (Private Branch Exchange) used for voice and data is called "Third generation digital PBX". It switches digitized voice and data.

The Data Module (DM) shown is used to connect both a telephone and a modem to a single line. The same kind of DM is used between the HP3000 ports (ADCC or ATP) and the PBX. (This is shown in the upper part of the illustration.)

In choosing the PBX as an office information network, business are making certain tradeoffs based on the advantages/disadvantages of the PBX.

An obvious PBX advantage is the use of existing wiring. This approach allows for flexibility in moving workstations and integrating voice/data communications. A disadvantage of using a PBX for digital communications is its cost. It costs an average of \$1000 additional dollars per line.

HP already has an aggressive PBX certification program to meet the requirements of those customers who choose to use the PBX as their office network. The goals of the certification program are to reduce the interface costs and increase service and integration to PBX-oriented customers.

Although HP does not intend to do its own version of the PBX, we have a high speed, PBX-to-computer interface call DMI (Digital Multiplexed Interface). The data module is no longer required on the host side, lowering the cost of PBX to HP 3000 connections. HP DMI currently interfaces to AT&T's PBXs and will work with other vendor's PBXs in the future.

DMI represents a growth path with total software compatibility for the current configurations of ATP's and ADCC's for the HP 3000. It provides the customer with 23 data channels at 64 Kbps each, which means that it replaces 23 DMs providing lower cost and higher throughput.

A new ATP, ATP for Meridian SL-1 interface, provides an RS-422 connection between the HP 3000 and the Northern Telecom Meridian SL-1 PBX. This connection also eliminates the data module on the host side, lowering the connectivity cost.

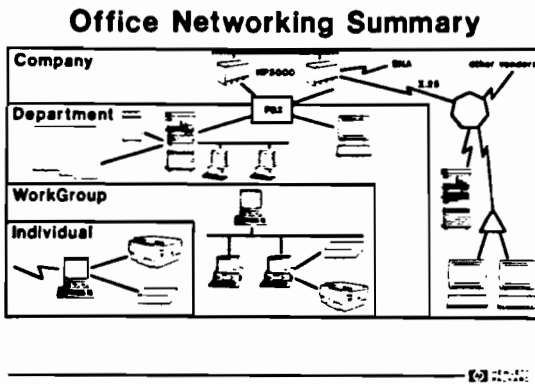
Another step to reduce the cost of the connection to the PBX is replacing the DMs on the terminal to PBX lines. Within the framework of ISDN, a standard is emerging for an interface between the telephone and the terminal that will connect the terminal to the PBX at 64 Kbps.

KEY POINTS

- PBX emerges as a cost effective solution for workstation to system connection.
- New technologies will further decrease the cost per connection.

OFFICE NETWORKING

SUMMARY



Networking in the office involves individuals, workgroups, departments and the whole company.

Each of these need voice, data and text (mail) communications and information access and sharing.

HP AdvanceNet satisfies these needs.

The individual PC can communicate to his work group, department, company and external services using an async (RS232) port and/or an IBM 3278 port. The workgroup can share resources and communicate using OfficeShare.

The department can share high cost peripherals (discs, laser printers) and access data bases on the departmental HP3000 using OfficeShare and the Disc Central and Print Central capabilities.

In a site network, a LAN based on IEEE 802.3 can interconnect departmental systems. If terminals need to communicate to multiple systems, a PBX is a viable solution. To network multiple sites, an X.25 based private network is a cost effective solution which provides multi-vendor communications.

HP 3000 SNA products provide communications between HP and IBM systems.

KEY POINTS

- HP AdvanceNet provides networking solutions for the individual, workgroup, department and company wide communications and resource sharing.

MANUFACTURING OPERATIONS ENVIRONMENT

TRENDS

The Operations Environment



Trends:

- Focus on quality/productivity
- Flexible manufacturing
- Multiple vendor equipment

* FOCUS ON QUALITY/PRODUCTIVITY

In the 80's the awareness of quality grew significantly which impacted automation and need for measurements and processing in every phase of production. An immediate feedback to errors on the production line is one example for productivity and quality improvements.

The quality issue and the use of new technologies (robotics) can increase competitiveness of manufacturer.

* FLEXIBLE MANUFACTURING

Another trend is of customizable manufacturing which is small production runs each with unique options. This kind of manufacturing needs more timely access to information, tighter control and frequent scheduling. These factors led to a trend towards fully automated factory floors.

* MULTI-VENDOR

Three levels of automation commonly exist:

- On the first level we see programmable controllers and numerical control from vendors like Allen Bradley, Gould, Modicon, etc.
- The second level is the supervisory control. HP, DEC and others provide systems for this level.
- The third level is plant, division and corporate information systems, who tend to use mainframes like IBM or general DDP systems like HP 3000.

Some plants have more than 100 vendors supplying production equipment. To increase productivity through increased in-plant efficiency, various levels of automation need to be linked and use consistent and up-to-date data.

MANUFACTURING OPERATIONS NEEDS

The Operations Environment



Requires support for:

- Multi-vendor communication
- Voice/data/video
- Guaranteed response time
- Factory environment

* MULTI-VENDOR

As discussed before, the existence of multiple vendor equipment in the various levels of control needs a multi-vendor network based on standards.

* VOICE/DATA/VIDEO SHARING FACILITY NETWORK

Many manufacturing companies like to use one network on the factory floor both for data, voice communication, and video broadcasts. There is a trend toward using the same network as the site-wide network that facilitates also engineering and office needs.

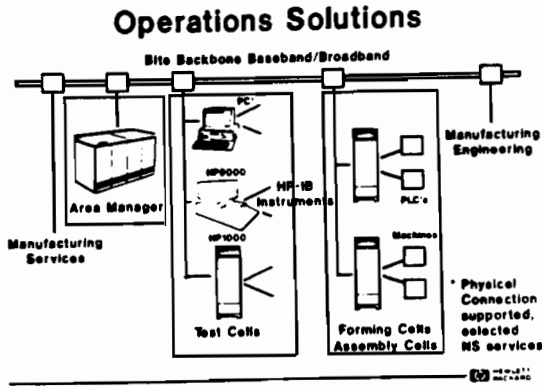
* GUARANTEED RESPONSE TIME

The production machinery needs guaranteed response time, since it relies on consistent and repetitious processes.

* FACTORY ENVIRONMENT: ROBUST AND LONG DISTANCES

Factories have unique networking requirements. Given the physical environment, the network must be environmentally robust. Factory floors of major manufacturing companies tend to be large (over 500,000 square feet), so long distance cabling is needed.

MANUFACTURING OPERATIONS SOLUTIONS



A site backbone LAN is a cable which runs through a site or a building and provides utility for data (baseband) or for data, voice, text and video (broadband). A site backbone can be baseband (one channel) or broadband (i.e. the band is split into several channels). Broadband also provides longer distances and better immunity to harsh environments than baseband. Subnetworks branch out of the facilities LAN. The network shown for test cells and assembly cells is a baseband IEEE 802.3 network.

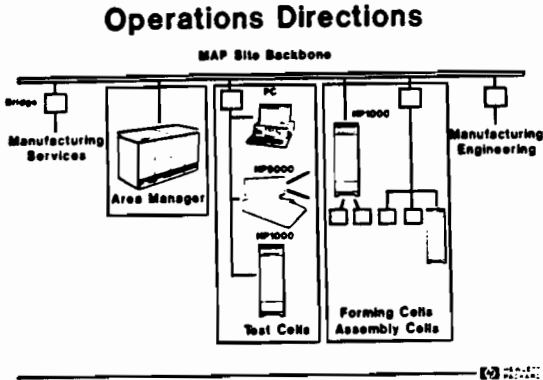
HP Network Services (NS) is the higher level software that provides file transfer between PCs, HP 1000, HP 9000 (HP-UX) and the HP3000 or DEC VAX acting as an Area Manager in Manufacturing Services.

To connect baseband subnets to a broadband backbone cable, a repeater is needed. HP offers a repeater today will offer a wiring distribution center which will connect up to 4 subnets to a backbone link.

KEY POINTS

- The backbone link for manufacturing facilities can be either baseband (using IEEE 802.3) or broadband.
- A backbone cable ties thin LANs subnetworks from assembly forming and test.
- HP Network Services provide a Network File Transfer capability between cell controllers (HP 1000, HP 9000 and PCs) and the Area Manager (HP 3000).

MANUFACTURING OPERATIONS DIRECTIONS



As a result of the various requirements of factory environments, there have been many industry approaches to factory networks. Many customers have developed their own highly specialized networks; Consequently there is little definition of an industry standard factory network. Installed broadband networks are often seen in this environment, but are not universal.

General Motors, HP and other vendors are working together to develop a factory network that is based on standards. The software that drives this broadband network is called MAP (Manufacturing Automation Protocol). The software is standards based, and has the potential of becoming an industry standard for factory networks.

Building on this foundation, HP plans to offer factory networking solutions based on the MAP standard, implemented on a broadband cable.

In general, we see factory local area networks being tied into facility-wide local area networks. The factory LAN will tie together the dedicated controllers and supervisory control systems needed to perform the first and second levels of factory automation.

Since the factory production systems must communicate with both the factory-unique systems as well as the company's business system, HP will provide the bridges for tying all these networks together.

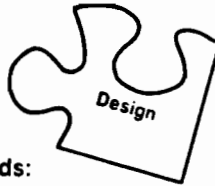
KEY POINTS

- General Motors, HP and other vendors worked together to develop a networking solution for manufacturing.
- MAP which uses broadband 802.4 LAN and other standard protocols, has the potential of becoming an industry standard.

THE DESIGN ENVIRONMENT

TRENDS

The Design Environment



Trends:

- Decreasing design cycle times
 - Simulation replacing build/test
 - Emergence of workstations
 - UNIX as standard OS
 - Increasing use of PC's
-

* DECREASING DESIGN CYCLE TIME.

This trend is a result of shortage in engineers, rising labor costs and increased competition. Design productivity is a solution to these issues and it means tools for decreasing design cycle time.

* SIMULATION REPLACING BUILD/TEST.

This is another design productivity tool. The outcome of this tool is increased design sophistication and systems' loads.

* EMERGENCE OF WORKSTATIONS.

In the past engineers used terminals connected to a mainframe. This configuration which did not provide the productivity and flexibility requirements of many organizations led to the workstation concept, like the HP9000, on each engineer's desk.

Two key application areas are Computer Aided Design (CAD) and Computer Aided Testing (CAT). Typical applications are finite elements analysis, printed circuit board layout and design, simulation drafting, modeling, etc.

* UNIX AS A STANDARD OPERATING SYSTEM.

Unix is widely used today in engineering environments, and is implemented on a variety of systems from mainframes to microcomputers. Many third parties base their application

software on Unix-based machines. Software development engineers and scientists like the tools and environment that Unix provides.

* INCREASING USE OF PCS.

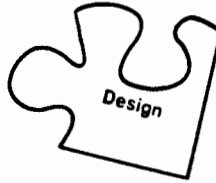
The emergence of PCs had its impact on the design environment too. PCs became the main office tool. Since many software packages developed for PCs are also useful in the engineering office, it had made sense to add low end CAD applications to those PCs to serve design needs.

KEY POINTS

- Shortage in engineers required increased productivity by reducing design time and using simulation instead of build/test.
- High performance workstations is one way to increase productivity.
- Unix is widely used on mainframes, minis, PCs, and workstations.
- PCs started to show up in the engineering office for low end CAD.

THE DESIGN ENVIRONMENT NEEDS

The Design Environment



Requires support for:

- Clustered workstations
- Links to PC's, minis, mainframes
- High speed data transfer
- Resource & data sharing

* CLUSTERED WORKSTATION

As mentioned earlier, high performance workstations like the HP9000 are replacing multi-user systems. The engineers work group forms a cluster of workstation with the following attributes.

* LINKS TO PCS, MINIS, MAINFRAMES.

PCs are used for low end CAD. Minis are used as computational servers. Mainframes are used mainly for integration of design phases, storing design libraries, and processing CPU bound calculations. Since all of these are part of the design process, links between these components are needed. This calls for standards that will be used by the different vendors providing these components.

* HIGH SPEED DATA TRANSFER

Engineering design involves large volumes of data. Engineers working on a common design need to communicate with each other and with mainframe applications to transfer large files of data. These transfers require high speed links.

* RESOURCE SHARING

The sophisticated and expensive peripherals needed in this environment are not fully utilized by a single engineer. Sharing of peripherals requires high speed and transparent access to files, data bases and peripherals.

KEY POINTS

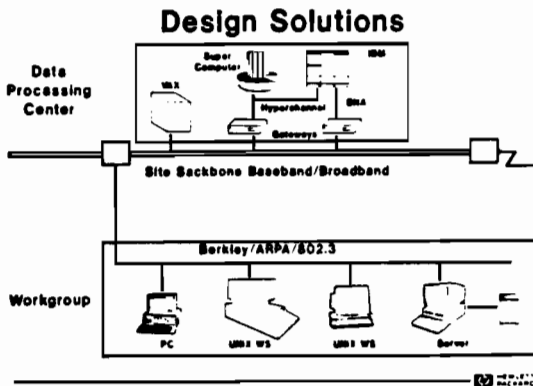
- Engineering work group need clustered workstations.

- PCs, minis and mainframe are components used in the design process.
- Within a work group high speed transfer of large files is needed.
- Expensive peripherals have to be shared within the work group.



THE DESIGN ENVIRONMENT

SOLUTIONS



In the engineering design and test market, HP's primary focus is to provide high performance local area networks. This high performance LAN provides the capabilities required for this environment. Workstations communicate with each other quickly and share resources (i.e., files, data bases, and peripherals).

The engineering design backbone LAN can be either baseband (IEEE 802.3) or broadband depending on the facility size and communicating needs. The backbone ties in subnetworks which serve design workgroups. The workgroup LAN is LAN/9000 which is IEEE 802.3 compatible and it supports HP 9000 series 500, 300 and 200 UNIX workstations. The Network Services provided between the workstations are Network File Transfer, Remote File Access, Remote Process Management, InterProcess Communications and Level 2 Link Access.

Under HP AdvanceNet NS, NFT is available between HP9000 and an HP3000, HP 1000, or Dec VAX. This allows communications between engineering/design, manufacturing operation and manufacturing services.

ARPA & Berkley Services will provide communications to other vendors. UNIX workstations supporting Berkley/ARPA/802.3 protocols. UUCP and CU UNIX services are currently available. These protocols are especially useful in communicating with computer systems and workstations from other vendors.

As mentioned earlier, there is a need to share expensive peripherals among several engineering workstations. The Shared Resource Manager (SRM) and a HP 9000 acting as a UNIX server offer this capability today. The SRM is the primary network for HP 9000's running the BASIC or PASCAL operating systems and it allows users to mix HP 9000's running different operating systems in one system.

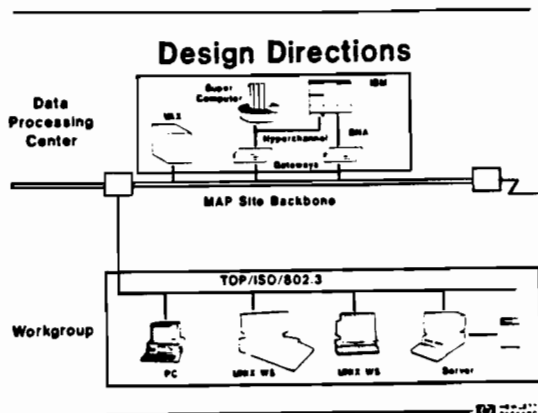
To communicate with DEC/VAX/VMS, we offer an HP AdvanceNet NFT.

On top of these capabilities, various terminal emulations are available for IBM and DEC communications.

KEY POINTS

- IEEE 802.3 LAN is the high performance link for clustered workstations.
- IEEE 802.3 backbone LAN is used to link to the other environments.
- SRM enables resource sharing for BASIC/PASCAL workstations.
- ARPA/Berkley are key to multi-vendor communications.

THE DESIGN ENVIRONMENT



The main change that we see for the future is in the protocol implementations:

- In 1986 the Network Services were HP proprietary NS and the defacto industry standard ARPA/Berkley.
- In the future, we foresee solidification of the Technical Office Protocol (TOP) as the standard for higher level services in the engineering environment.
- ISO levels 3-4 will replace or coexist with ARPA's TCP/IP as the transport and network protocols. This migration will occur gradually with the demand for ARPA/Berkeley Services remaining strong for the future.
- The backbone and broadband link with Manufacturing Automation Protocol services seem to be the dominant protocols for a manufacturing network.

MULTI-ENVIRONMENT INTEGRATION

HP AdvanceNet provides you with integration of the manufacturing, engineering, and office environments by providing Network File Transfer service between **HP 3000**, **HP 9000**, **HP 1000**, and **DEC VAX/UMS** systems. Additional integration is provided by **PC** to **HP 3000** communication and **HP 3000** to **IBM** communication.

HP ADVANCENET BENEFITS - TODAY

HP AdvanceNet Benefits

Range of solutions to meet your needs

- Improved productivity & quality of service
 - Information available where needed
 - Information easily accessed
 - PC integration
 - Electronic mail
- Reduced cost
 - Connectivity options
 - Sharing resources
 - Common network services



The solutions are available today! There are different alternatives that provide both local solutions (PBX, LAN and direct connect) and remote solutions (dial up, leased lines, public or private data networks). The solutions allow you to communicate between the various work environments, where different HP systems or other vendor systems might exist.

The benefits of our networking solutions is improvement in your company quality of service, productivity and the reduction of costs of operations.

* IMPROVED PRODUCTIVITY AND QUALITY OF SERVICE

Today businesses face the challenge of providing high quality customer service while remaining competitive and profitable; i.e., increase efficiency.

In the last decade Distributed Data Processing proved to enhance productivity and quality of service by providing up-to-the-minute information. Personal computers increased it even more, bringing the processing power closer to the end user, decreasing response time while being linked to the larger systems for further processing.

Electronic mail improved the quality of communications and made it easier to transmit text, data files, and graphics within an organization and outside.

One of HP AdvanceNet's objectives is to take the complexity out of networking. The details of the architecture structure and various link and protocol alternatives allow professionals to design a better and more cost effective network that is transparent to the end user. The end user interfaces to applications that use network services. Everything else within the network: Link technologies, the topology and protocols used are not seen by the end user.

* REDUCED COST

Implementing or enhancing your own computer and communication network provides economic benefits to your business.

First and foremost, it will help you drive down the cost of doing business today. We have already discussed how the use of standards reduces cost.

One of our major accounts used the various connectivity options to fit its geographical layout and reduce costs. He used a private X.25 based network for his main offices. For small offices which had low traffic, he used dial-up links into the private networks. In several remote locations where leased lines were too long and costly, he connected to a public data network which interfaced to his private network. The alternative of using leased lines throughout the organization was much more expensive.

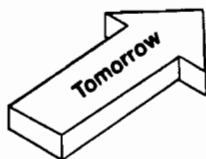
Shared resources reduce peripherals' cost. We provide shared resources in PC networks, engineering networks (SRM) and distributed systems networks (through remote peripherals access and network spooling).

Another aspect of shared resources, is sharing computer resources. Results produced on one system can be transferred to another system. Personal computers can share the data bases, applications and peripherals of larger systems.

The common network services across HP systems make it possible to use the right system for a given work environment and communicate between the various work environments. For the example of a payroll application, suppose you need to get timesheet data from a HP 1000 and benefit records from the corporate HP 3000. HP AdvanceNet common network services will let you transfer the files or records from the HP 1000, to the departmental HP 3000 and access the benefit records in the corporate HP 3000.

HP ADVANCENET BENEFITS - TOMORROW

HP AdvanceNet Benefits



Protects Your Investment

- Flexible architecture
- Based on standards
- Backward compatible

Networking is considered by many managers as a complex issue. The terminology is very technical, the alternatives are numerous and making mistakes when choosing a networking strategy might turn out to be costly.

HP recognizes this challenge and that is the reason that we have designed a networking architecture that will protect your investment.

The flexibility of the architecture to add standards (and not just replace them) in the future when they become available, ensures that by making a decision today, you will not encounter a dead-end in the future.

The fact that our implementations are based on standards means that our solutions are stable and have a long useful life. It was always HP's strategy to protect its customers' investments by providing backward compatibility in addition to new enhancements. With HP AdvanceNet we proved it once again: Compatibility is kept with the HP/DSN/DS network services; additionally, "old" HP/DSN/DS nodes can be part of an HP AdvanceNet network.

SUPPORT SERVICES

Hewlett Packard assures you support throughout the life of your network. The HP Network Support Program will be soon be implemented in the U.S., Canada, and parts of Europe. The program will offer a full range of services including planning and design, installation, maintenance and education.

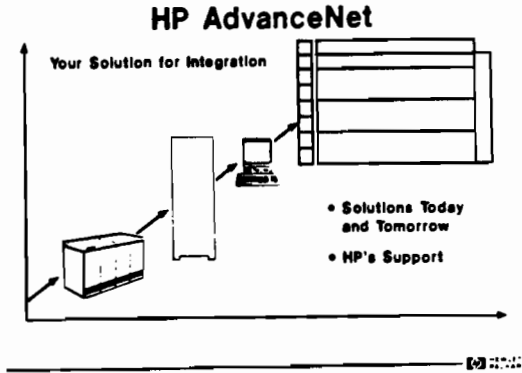
HP can help define a network strategy to meet a business' goals and objectives. Experienced HP network consultants are available to analyze a business's communication needs and design a network to meet those requirements.

Recognizing the multi-vendor environment of networks, HP has services to customize the installation of the network into a business' day-to-day activities. HP also provides support services to isolate and resolve problems on networks which are composed of HP and other vendors equipment.

HP's field organization, consisting of Network Consultants, AEO, CEO, technical centers, project centers, and response centers, are highly qualified and backed up by extensive resources.

We have sales and support offices in every major area of the world, this is especially important in a wide geographic area network that needs support close to every site.

YOUR SOLUTION FOR INTEGRATION



Hewlett-Packard's new generation of networking - HP AdvanceNet, is a strategy, an architecture and a series of products that WORK.

Our first networking product was delivered thirteen years ago, and currently we have more than 30,000 systems installed in customer networks throughout the world. That's proven experience!

We have shown you today that we have the solutions today. We know that choosing a networking strategy is a hard decision. You can start your network with a few nodes and easily add more nodes as you grow.

We have shown you that HP AdvanceNet provides the architecture and products for integrating the various work environments into an Integrated Information Management System.

HP AdvanceNet is more than a promise; it is a reality! HP AdvanceNet provides a multitude of network services within an HP computer family, between HP computer families, and to other vendors equipment (including DEC and IBM).

Please join us at the HP booth. We are proudly demoing our intersystem connectivity. Network File Transfer is available TODAY across HP 3000s, HP 1000s, HP 9000s and DEC VAX.

ISDN Networking for the Office

Tim Shafer
Hewlett Packard Company
8000 Foothills Blvd.
Roseville CA U.S.A. 95678

This paper is a consideration of Integrated Services Digital Networking (ISDN) and its significance to people who use and manage business and office networks. Those of us who have seen a growing influx of mail touting a dizzying array of dissimilar "ISDN" products will benefit from this simple but insightful look into ISDN.

Although this paper focuses on ISDN with regard to office networking, it should be noted that ISDN also will have an important impact on wide area networking. ISDN standards are expected to be particularly helpful in standardizing the way in which telecommunications is used to connect together local voice and data networks.

In the office, ISDN is facilitating the expansion of office networks by allowing terminals or personal computers to plug into the network anywhere there is a telephone line. The expansion of a manufacturing plant's office network through the connection of remote PC workstations to the network is a good example. Another example is the expansion of a sales office's network as portable computers and cellular telephones are used to connect the salespeople in the field to the office network.

ISDN as Evolution

Integrated Services Digital Networking (ISDN) is a term which has become synonymous with the evolution of our telecommunications systems from the traditional analog speech grade lines of the past to the digital circuit, packet, and wideband switching networks of the future. Naturally, these future networks will be called ISDNs. The process of ISDN evolution promises to reduce the price and improve the performance of communications facilities and services we are already familiar with, while at the same time opening the door to valuable new technologies.

Reshaping our telecommunications networks into tomorrow's ISDNs will be a long and expensive evolution driven by the growing need for data communications. Data communications, not voice, will fuel the demand for ISDNs because plain old telephone service is a low growth business with little to offer the business customer looking for a way to become more competitive using telecommunications (Standard & Poors estimates 2-3% growth in telephone access lines during 1985). Data communications, on the other hand, is experiencing relatively strong growth in demand which is likely to increase with the accelerating development of applications and management tools for computer communication (Data Communications

1982-1985 economic index shows roughly +9% average growth per year in data communications market).

Nowhere is the need for our telecommunications networks to evolve toward ISDN more evident than at the network interface. Most of us, although armed with our company's modern private branch exchange (PBX) telephone system, have fumbled hopelessly with various cryptic key sequences on our touch-tone telephones as we tried to transfer a call or invoke one of the other 150 features provided by the PBX. ISDN goes along way toward eliminating this problem by providing interfaces capable of supporting computerized terminals which provide a much more friendly interface between the user and the network.

Within the telephone network itself, the need for ISDN is clear. Today's telephone switching systems are digital and are controlled by computers running sophisticated programs and requiring the ability to efficiently (via packets) exchange data in order to provide new services and network management.

In the office, personal computer users are experiencing a growing need for economical data communication that is pushing them toward the use of existing, low cost telephone wiring (twisted pair) systems for digital data communication. Because this type of wiring system can be used to connect computer terminals everywhere there is a telephone (without disrupting telephone service) it offers great flexibility by allowing terminals to be as portable as telephones. Many of us will gladly forget about coaxial, twinaxial, and other more bizarre forms of computer cable and turn instead to uniform wiring systems based on twisted pair cable.

While the long term course and effect of the ISDN evolution are unpredictable, we clearly seem to be headed toward some important and foreseeable goals during the next ten years. Low level integration (physical integration) of data communications onto telephone networks is a significant first step in the ISDN evolution and it lays the foundation for functional integration, which is the second crucial step.

Physical Integration of Networks Cuts Cost

In the future, businesses experiencing a growing need for diverse data and telecommunications will not be able to afford the expense of installing and managing multiple wiring systems in the local environment. As labor costs increase while computer and communications cost versus performance drops, it will become more cost effective to integrate telephone and data networks onto the same wiring system in local environments (see figure 1). Twisted pair telephone wiring is the predominant form of wiring found in office environments and has many favorable attributes, including relatively low cost and the ability to support local 1-3 Mbps data transmission, making it the best choice today for ISDN networking in the office.

The first, physical integration stage of ISDN evolution only provides users with the limited advantage of having local area network (LAN), RS-232, and telephone network together on the same wiring system, but not

switched through the same network in most cases. Building on the foundation provided by physical integration of these different networks, computer and communications manufacturers will be able to functionally integrate the networks, switching the various types of communication through a common network.

Functional integration of computer and telephony networks will be the second milestone in the ISDN evolution, allowing multimedia (voice, data, etc.) networked services. An example of such a service would include a data call placed from a personal computer, but involving a real-time voice input from the person initiating the call, in order to pass a security screening. One could imagine a stock broker using this service to initiate a financial transaction.

Many other illustrative examples of valuable applications requiring functional integration of multi-media networks can be imagined and most would rely on the use of a desktop computer to automate network access and call setup (dialing), directory assistance (remote data base inquiry), and the use of different media (voice, data, facsimile, etc.) during a call. Ideally this would mean your personal computer deals with the network and other computers connected to the network, and then lets you know when it has found the person or information that you requested.

Separating the Useful ISDN from the Not So Useful

The bad news is that we are not yet at the functional integration stage of ISDN evolution, and the broad definition of ISDN as an evolution provides no basis for distinguishing between useful ISDN developments (meeting user/manager needs) and ISDN garbage (a matter of perspective). The good news is that valuable progress has been made in the physical integration stage of ISDN, and the same user concerns that have traditionally been the market's flame test for networking offerings, apply just as well to new ISDN developments.

Multi-vendor compatibility, network management, flexibility, and cost frequently appear at the top of the list of user/manager concerns when considering new networking products. Although these are all applicable to ISDN developments, a general consideration of each of these with respect to ISDN goes beyond the scope of this paper. A look into standards for ISDN is important, however, because significant work in defining these standards has been done and will be useful in addressing many user concerns. User's evaluating new communications equipment can benefit by understanding something about the ISDN standards the capabilities of conforming equipment. Manufacturers, in turn, should be able to provide information on ISDN standards and their equipment's conformance.

ISDN as a Emerging Set of Standards

Committed to promulgating the international compatibility that we have come to expect for voice communication via telephone, the Consultative Committee for Telephone and Telegraph (CCITT), a branch of the United Nations, has made great progress during the last ten years in developing

standards for ISDN. Countries participating in the CCITT (the U.S. is represented through the State Department) usually adopt its recommendations (CCITT terminology for standard) as domestic standards. Since manufacturers typically have the greatest expertise in technical areas addressed in the standards, they funnel their input into the CCITT via domestic standards organizations (Exchange Carriers Standards Association in the U.S.).

The CCITT is organized into study groups which attempt, during a 4 year study period (the last period ended in 1984), to develop recommendations which resolve specific questions of interest to participant countries (e.g. how to develop a uniform numbering system for voice and packet data calls). In the past, the CCITT has developed many of the voice and data communications standards we are familiar with today, including X.25, V.24 (RS-232), G.703 (T1), and others. Study Group 18 of the CCITT is specifically responsible for creating standards for ISDN, which in some cases are developed from scratch and in other cases are simply adopted from pertinent recommendations already developed by other study groups.

Without going deeply into the work done by Study Group 18 or domestic standards organizations, it is significant to note that most of the progress made so far has been in the definition of a limited set of new interfaces through which users may attach to an ISDN, and the definition of adapters which will allow existing digital interfaces to connect to an ISDN (see Figure 2).

The 2 new ISDN interfaces which are most important to business/office network users are the Basic rate and Primary rate interfaces. These interfaces, along with adapters to support RS-232, X.25, and other existing protocols, are already beginning to appear on new telecommunications switching systems being used by the Regional Bell Operating Companies in ISDN trials. It is expected that within the next three years ISDN interfaces will also appear on new PBXs, which provide the lion's share of business telephone access lines.

Both the Basic and Primary interfaces are built using 64 Kbps channels for the transport of user data (called B channels in CCITT lingo). Although the 64 Kbps rate has been in use in the U.S. prior to ISDN, 64 Kbps channels have typically been restricted to carrying only 56 Kbps of computer data. The restriction stemmed from the network's need to send information to control the call (called signaling information, it is used to invoke call set up, tear down, etc.) in the same channel used for the call itself. Digital voice is not noticeably corrupted by the network's periodic insertion of control bits into the data stream, but computer data must be sent in a special 56 Kbps pattern to avoid corruption by the network. For this reason, and the fact that this old method of signaling is only capable of sending the most rudimentary information, it has been scrapped in the move toward ISDN. Replacing the old method of signaling, a new protocol referred to as Common Channel Signaling has been defined by the CCITT.

Basic rate provides 2 of the 64 Kbps data channels and is thought to be most suitable for desktop terminals. Primary rate carries 23 of the B channels and provides a good means of connecting host computers to an

ISDN. Both interfaces have a single, "Common" channel used to carry all the signaling information pertaining to all of the calls taking place on the interfaces B channel. The use of a common signaling channel benefits user's in 2 ways; first, it frees the B channels so that they can be used for unrestricted 64 Kbps transmission, second, it supports more complex information exchanges between the subscriber terminal and the network in order to facilitate more effective network management.

Progress in defining the Basic and Primary rate interfaces has been from the wire up into the protocols to be used across the interface on the B and D channels (starting at the Physical Layer of the Open Systems Interconnect description of the interfaces). Primary rate will use an existing, previously CCITT standardized form of the North American T1 transmission system (1.544 Mbps) which provides 24 of the 64 Kbps channels (23B + 1D) which are time division multiplexed onto a 4 wire, point-to-point interface.

Unfortunately Basic rate does not have the virtue of being derived from an existing telecommunications standard. Instead, the Basic rate is a unique, 192 Kbps link supporting 2 B channels and a 16 Kbps D channel. Basic rate links can be configured in either a point-to-point or point-to-multi point configuration with the network on one side of the interface and 1-8 terminals on the other.

Work on standardizing the protocols to be used across these interfaces is incomplete, but most of the progress that has been made has focused on satisfying 3 requirements. The first requirement is to interconnect existing terminal and computer devices across the ISDN. The second is the need for a new and more versatile packet switching protocol to supplement the use of X.25. Finally, the third is the need for a highly functional signaling protocol for use in D channels.

Now that we are equipped with a basic understanding of the progress being made in ISDN standards, it is worthwhile to consider an example of how manufacturers are using these standards to help meet network user/manager needs. Further insight into ISDN standards is available from several good communications industry consultants (ea. Omnicom Inc. 703-281-1135).

DMI is an Example of ISDN Development

The new Digital Multiplexed Interface (DMI) is a good example of ISDN development for office networking because it exemplifies the physical integration stage of ISDN evolution, it is based on the ISDN Primary rate standard, and it is widely supported by computer and communications manufacturers. Today, 66 companies publicly endorse the use of DMI while over 100 have licensed the specification from AT&T (see Figure 3). AT&T developed the DMI specification in 1983 after recognizing that the emerging ISDN Primary rate could act as the basis for a new multiplexed RS-232 interface useful in connecting PBXs and host computers (computers serving terminals).

AT&T accurately predicted that DMI would reduce the cost and complexity of using a PBX to provide switched RS-232C networking via the PBX and its

network of telephone wiring. Cost and complexity reduction would come from the use of DMI to replace expensive, discrete RS-232 connections between the HP 3000 and the PBX (see Figure 4). AT&T also accurately predicted that many computer and communications manufacturers would be attracted to DMI because it could be implemented in a way that would allow its capabilities to grow gracefully (via software enhancements) as more of the ISDN standards emerged.

Although DMI by itself is not a dramatic example of ISDN's physical integration, since it does not carry voice and data, it is an important step in bringing users toward convenient and cost effective voice and data networking on a single telecommunications system.

DMI was not the only industry effort to utilize T1 as a multiplexed RS-232 connection between host computers and PBXs, but it is the only industry effort to implement the Primary rate interface, and AT&T has committed to evolving the specification in accord with CCITT enhancements to the Primary rate. Northern Telecom Inc.(NTI) and Digital Equipment Company (DEC) introduced in late 1984, a similar product known as the Computer to PBX Interface (CPI). DMI and CPI, although functionally similar, are differentiated in that CPI was developed with the emphasis placed on backwards compatibility (supporting non-ISDN compatible T1 with 56 Kbps data channels and old fashioned signaling). DMI, on the other hand, was developed as a more forward looking architecture and relied on the newer Primary rate compatible version of T1.

The different approaches taken by the CPI supporters and the DMI supporters were an inevitable consequence of the inability of manufacturers to cost effectively support both backwards compatibility and new capabilities. Fortunately, Northern Telecom has publicly stated its intention to develop a Primary rate compatible version of CPI in the future, and manufacturers implementing DMI may also be able to connect to the new CPI.

Multi-Vendor Compatibility Requires more than Standards

The DMI User's Group, an association of companies licensing the specification, gradually grew during 1985 and 1986. Through its meetings every 4 months, the User's Group became an important forum for the sharing of information related to development, compatibility testing, etc. It is noteworthy that in many cases in the past, manufacturers without the benefit of an industry forum such as the DMI User's Group have developed incompatible implementations of the same standard (RS-232 is a great example).

DMI was demonstrated by HP and AT&T at Interface 1985 using an HP 3000 minicomputer and a System 85 PBX. In late 1985, AT&T introduced its DMI product for the System 75 and System 85 PBXs. HP became the first computer manufacturer to introduce a computer interface for DMI in May 1985.

The HP-AT&T linkage created by DMI approximately halved the hardware expense previously associated with establishing 23, RS-232C channels

between the HP 3000 and the System 75 or System 85. By reducing cost and complexity, DMI benefited user's by making flexible PBX networking more affordable while at the same time it helped network managers by eliminating multiple discrete connections. HP's DMI interface further addresses the need for easy management because it comes with extensive monitoring and diagnostic software, and because it has a fault tolerant architecture allowing damage to individual channels without compromising the performance of the remaining channels in the interface.

Completing DMI development was only part of the effort put into making the new DMI interface a success with HP and AT&T customers. HP and AT&T have conducted extensive certification testing of RS-232 configurations using DMI. Details on supported networking configurations using DMI are available from HP and AT&T. In addition to certification, HP and AT&T have announced the intention to provide a joint support service for customers using DMI between HP and AT&T equipment. HP-AT&T joint support service is aimed at helping customers receive fast, cooperative support from both manufacturers by telephoning either HP or AT&T.

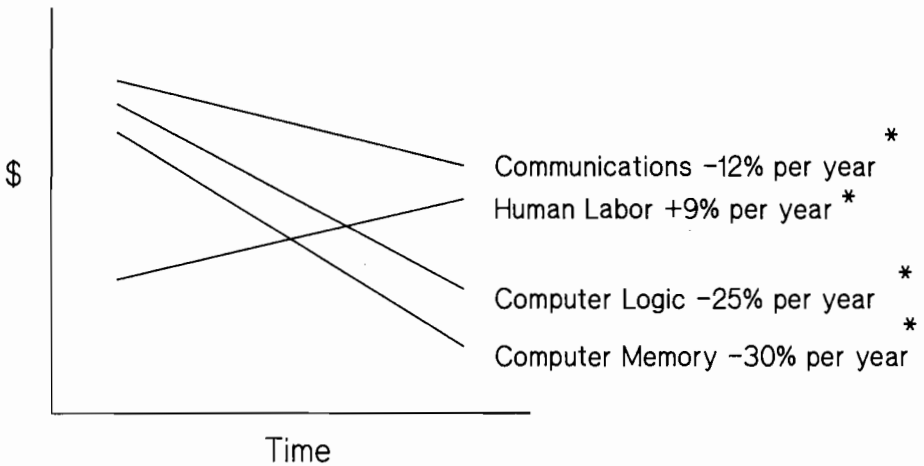
Positioning ISDN against Networking Alternatives

As ISDN evolves it's position changes with respect to other networking alternatives. Looking 3 to 5 years down the road, PBXs acting as ISDNs seem to be a good center for office/business networking, acting not only as the workhorse for most voice and data communication, but as a means of interconnecting local area networks, wide area networks, computers, and terminals. This view of ISDN within the office is well supported by the emergence of packet switches as PBX adjuncts, and the desire of many large customers (ea. Electronic Data Systems/General Motors) to use the PBX as a means of interconnecting LANs via a wide area network (see Figure 5).

The capabilities of non-ISDN networks will certainly continue to grow at the same time ISDN develops, and they will continue to meet user needs that are not well accommodated by ISDN. For this reason, ISDNs, LANs, and wide area networks should be considered complementary, and equipment manufacturers should address the customer's need to use and manage all these types of networking.

Figure 1

Integration of networks onto existing telephone wiring reduces labor costs associated with managing multiple networks

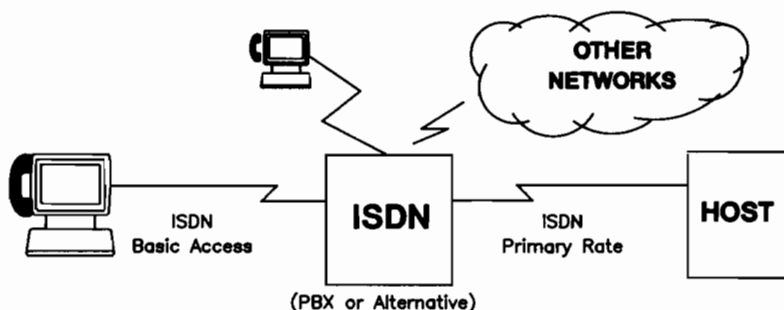


- Leverage from existing wiring cuts installation time/cost
- Telephone wiring is pervasive— increasing flexibility
- Telephone wiring is comparatively inexpensive

* (Trends derived from information gathered during last 15 years by the Bureau of Labor Statistics and electronic industry consultants)

Figure 2

ISDN Standards will Replace Proprietary Interfaces and Services on Telecommunications Networks



THE ISDN APPROACH

– Standard Interfaces

- * Basic Access and Primary Rate
- * Usefull for Computer-to-PBX
PBX-to-PBX, Computer-to-Computer
- * Mutiple Channels per Interface
- * 64 Kbps Voice or Data Channels
- * Single Channel in Each Interface
For Administrative Control of
Voice/Data Channels

– Standard Services

- * Digital Circuit Switching (Voice)
- * Digital Packet Switching (Data)
- * Network Management
- * Compatible with OSI Framework/Protocols

Figure 3

Companies Publically Supporting DMI Product Development

| | |
|----------------------------------|-------------------------------|
| Advanced Computer Communications | Jistel |
| Advanced Micro Devices | L P COM |
| AT&T | Micom Interlan |
| Amdahl | Mitel |
| Applictek | Mitek |
| BBN Communications | NCR |
| Bose Associates | NEC America |
| Burroughs | Nixdorf |
| California Microwave | Nokia |
| CASE Communications | Prime Computer |
| Compaq Telecommunications | Radio/Switch |
| Control Data | Raytel Systems |
| CXC Corporation | Rockwell International |
| Dallas Semiconductor | Scitec |
| Data General | Siemens Comm. Systems |
| Datapoint | Societe Anonyme de Telecom |
| Davox | Sonacor Systems |
| Digital Sound | Soron |
| Distributed Solutions | Spectrum Digital |
| DMW Group | Speech Systems |
| Edward K. Bower | Syntrex |
| Four C Enterprises | Tadiran Electronic Industries |
| Gandalf Data Ltd. | Tandem Computer |
| Gold Star Tele. Elec. | Tekelec |
| Harris | Telettra Spa. |
| Hewlett Packard | Thompson—CSF Telephone |
| Hitachi America | Timeplex |
| Honeywell | Tricom Tele. Consortium |
| Idacom | Ungermann—Bass |
| Infotron | Wang Laboratories |
| Intecom | Western Digital |
| Intel | Ztel |
| ITT | |
| Jeumont Schneider | |

Figure 4

Digital Multiplexed Interface (DMI) simplifies PBX networking of terminals to host computers.

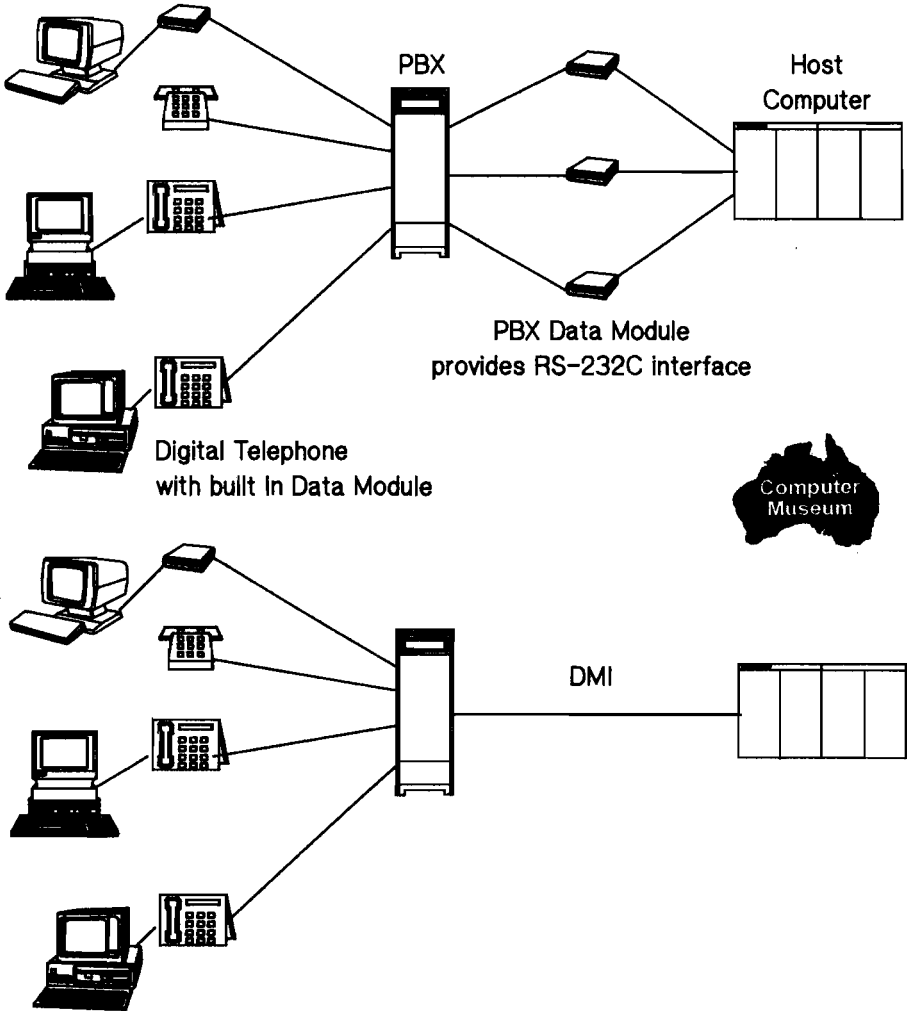
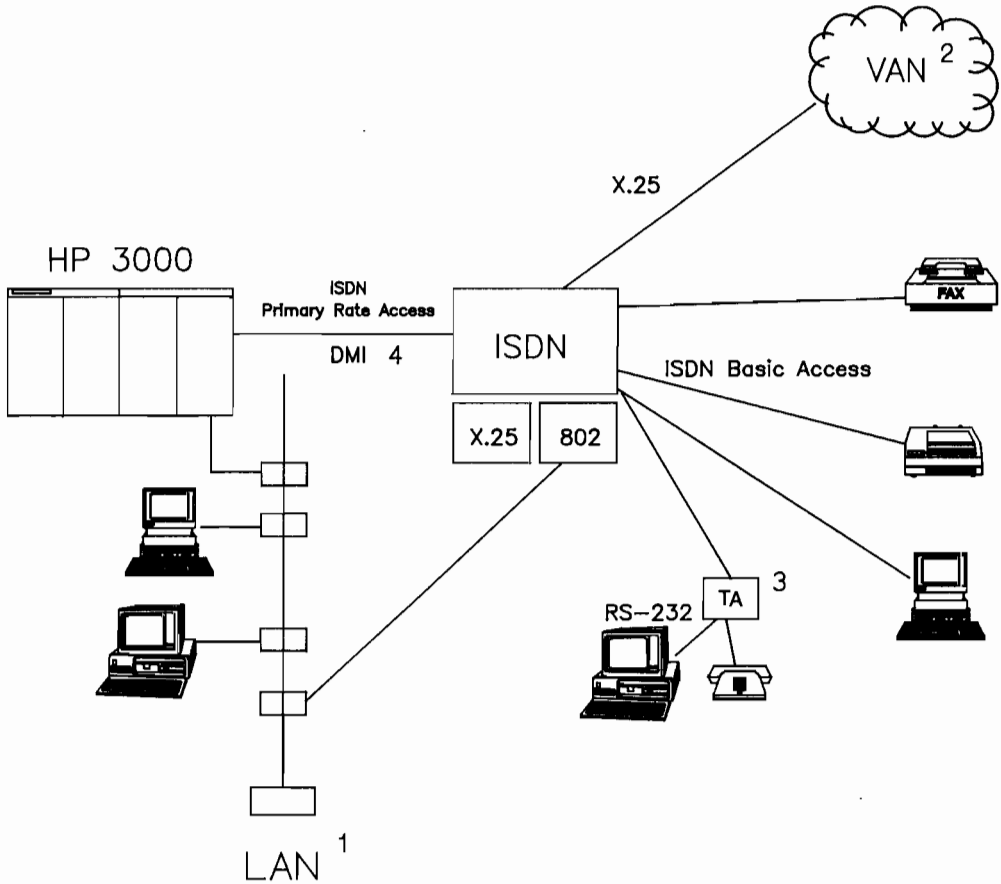
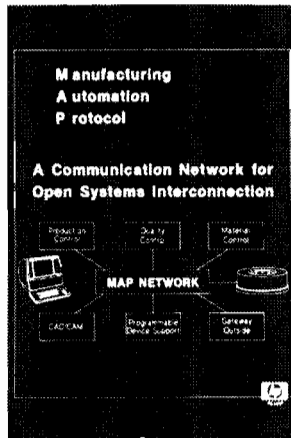


Figure 5
Future Distributed Office Communications



- 1 Local Area Network
- 2 Value Added Network or wide area network
- 3 Terminal Adaptor allows equipment without an ISDN interface to attach to an ISDN network

- 4 Digital Multiplexed Interface is development of the ISDN Primary Rate Interface



MAP Manufacturing Automation Protocol

Roberta Estes

Hewlett Packard
39550 Orchard Hill Place
Novi, Michigan 48050

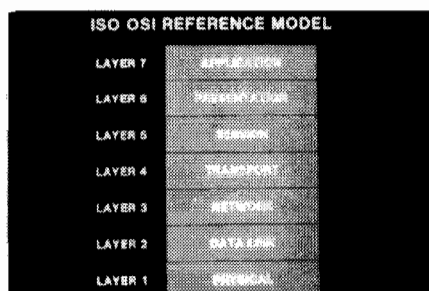
What is MAP?

Manufacturing Automation Protocol, commonly known by the acronym MAP, is a communication network standard spearheaded by General Motors beginning in 1983.

MAP is an evolving LAN (Local Area Network) standard designed to connect multivendor equipment in a factory environment allowing data sharing between equipment of different vendors.

MAP is based on International Standards Organization (ISO) reference model for Open Systems Interconnections (OSI) which provides non-proprietary access for any subscribing vendor.

The Open Systems Interface provides a standardized, seven layer architecture that modularly defines functionality for peer to peer device communications. Any vendor subscribing to the OSI architecture must provide the OSI defined functionality in each of the seven layers.



The Birth of MAP

General Motors has within its plants a widely diverse range of hardware and software components, virtually one of everything ever made. Most of these devices have limited connectivity, typically communicating only with other like devices or with selected vendors. Providing communications within a programmable controller to interface with limitless proprietary protocols is simply not feasible, so many factory floor equipment suppliers interface with one or two designated systems. Hewlett Packard's PCIF (Programmable Controller Interface) provides access to many proprietary programmable controller networks.

General Motors became painfully aware of these limitations and the accompanying expense of purchasing multiple host systems to control an ever growing number of intelligent manufacturing devices. The host could not be chosen for it's capabilities, but had to be chosen due to the communication capability with the installed programmable controller.

Finding this situation unacceptable, GM formed a Task Force to address the problem.

The MAP Task Force published a document calling for one standardized protocol, available to all vendors, allowing interconnection and sharing of data between devices within a factory.

The original MAP 1.0 specification was subsequently revised to level 2.0, and then 2.1. GM has set a goal of one standard level per year. Level 2.1 was agreed upon in March, 1985. Version 3.0 is targeted for April 1987. This allows the moving target to stabilize long enough for vendors to implement production pilot projects, companies to introduce products, and to provide a common point of reference. Changes found to be necessary to level 2.1 will be incorporated into level 3.0.

Versions 2.1 and 3.0 were originally intended to be upward compatible, however, scheduled changes indicate a conversion will be necessary. This incompatibility is caused in part by the emerging standards themselves that MAP uses as its foundation.

The most prominent example lies within the message passing facility within MAP version 2.1, MMFS (pronounced Memphis). MMFS provides a standard message format for all systems. MMFS was invented by GM due to the lack of any ISO standard addressing the issue. Today RS511 addresses this need and will be the message passing format for version 3.0 and upward. MMFS was an interum method to suffice while a standard was being written. MMFS and RS511 share many similarities, due to the MAP influence, but are not identical. MMFS messages will require recoding to be MAP 3.0 compatible.

OSI Reference Model Layers

The OSI reference model is broken into seven layers, each with a specific functionality that must occur within that layer, before proceeding upward or downward to the next layer.

Layer One - Physical Layer

Layer one is the Physical Layer which specified the actual media by which the data is transferred. It does not deal with how the data is transferred, for example analog or digital, but only with the physical transmission media itself.

Layer Two - Data Link Layer

Layer two, the Data Link Layer, manages data access to the cable. Within the IEEE 802 specification, node to node hardware error checking is also performed at this level.

The media access method choices are 802.3, CSMA/CD (Carrier Sense Multiple Access/Collision Detect), and 802.4 Token Passing.

CSMA/CD, specified for 802.3, uses contention to control access to the transmission media. Any node wishing to transmit listens for traffic on the cable (Carrier Sense), and can transmit if no traffic is detected (Multiple Access). Propagation delays may allow two nodes to begin transmitting simultaneously, and the transmissions collide. If a node detects a possible collision, it abandons the current transmission and attempts to retransmit after a brief random delay. Due to the real time factory production requirements, the CSMA/CD method was not chosen for MAP.

Token Passing, specified for 802.4, used a special bit pattern called a token to control access to the cable. The token travels from node to node continuously. The node with possession of the token has exclusive access to transmit its message to the network. If no data is to be transmitted, the node passes the token on to the next logical node.

Token passing is called a deterministic access method, versus CSMA/CD, which is referred to as a nondeterministic access method. An 802.4 network can be designed such that the transmission rate for a particular node can be determined exactly. For example, if 10 devices reside on a network, given equal priority, each device is assured of having the option to transmit every tenth time. In an 802.3 network, there is no way to accurately determine the amount of time a node might wait while other nodes transmit. Therefore, an 802.4 network can be modeled, in advance, to predict performance, while the 802.3 cannot.

Real time equipment must be guaranteed access to the transmission media. 802.4 is the Data Link choice for MAP. Token Passing is also the only Data Link protocol supported on broadband by IEEE 802. In addition, differing priority levels can be assigned to nodes on 802.4 networks. Many of the factory floor vendors are already somewhat broadband based, so less retrofitting is required than with any other media or data link choice.

Layer Three - Network Layer

Layer Three, the Network Layer provides addressing between nodes.

Layer Four - Transport Layer

Layer four, the Transport Layer is responsible for establishing connections between the data link entities. Additional error correction is performed at this level.

Layer Five - Session

Layer five, Session is concerned with dialogue management between the nodes connected through level four.

Layer Six - Presentation

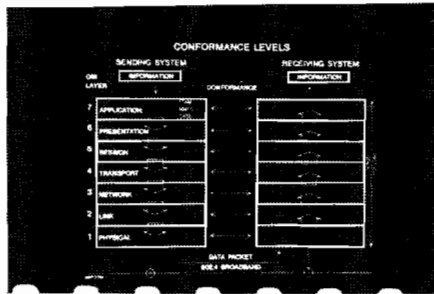
Layer six, Presentation, negotiates the languages spoken between the entities, i.e. ASCII, encryption coding, etc. MMFS (Manufacturing Message Format Standard) is the language of MAP, and is an interim language pending standardization of RS511.

Layer Seven - Application Layer

Layer seven, the Application Layer, provides services to the application. Within MAP, we have CASE, which makes and breaks connections programmatically, and FTAM which facilitates transfer of files. Virtual Terminal will eventually provide transparent terminal connections on remote nodes.

| MAP SPECIFICATION | |
|------------------------|---|
| LAYER 1 (PHYSICAL) | --- IEEE 802.4, TOKEN BUS ON BROADBAND OR BASEBAND |
| LAYER 2 (DATA LINK) | --- IEEE 802.2, LINK LEVEL PROCEDURES |
| LAYER 3 (NETWORK) | --- ISO NBS INTERNET PLANNED |
| LAYER 4 (TRANSPORT) | --- ISO NBS TRANSPORT PER CLASS 4 SPECIFICATION |
| LAYER 5 (SESSION) | --- ISO NBS SESSION PLANNED |
| LAYER 6 (PRESENTATION) | --- MAP DEFINED |
| LAYER 7 (APPLICATION) | --- FILE TRANSFER PER ISO NBS MAP VIRTUAL TERMINAL PER ISO NBS MAP DIRECTORY SERVICES PER MAP NETWORK MANAGEMENT PER MAP |

Conformance to the MAP standard is tested at each layer. Each communicating application must pass a series of tests, a scenario, that is evaluated by an application called a scenario interpreter. These applications assure standardized communication for each defined function within MAP at the specified level and in the accepted context. Each vendor aspiring to market a certified MAP product must retain a nonpartisan testing facility to certify their particular implementation of MAP applications.



Stages of an ISO Standard

The MAP levels, revisions and working documents seem very confusing. To an outside observer, it might appear that MAP implementors are chasing a continually moving target. Well, that observer would be right. Why would we do that?

As we discussed earlier, the final version of MAP will conform entirely to ISO standards. These standards, however, do not emerge overnight, and, similar to passing a law in the legislature, have a well defined path that results in several revisions.

The Working Draft is the birth of a standard. At this point, the majority of the content is open to change. Much debate will be occurring as to the real and theoretical benefits and disadvantages of the particular subject.

The second stage of the standard is the Draft Proposal. This usually occurs after some amount of research has been performed. The majority of the content is still open for change and will be modified as pilots become operational throughout the world.

The Draft International Standard, the third stage will not occur until the final form of the Draft Proposal has been successfully implemented repeatedly. At this point the content of the standard is considered to be stable, with no major technical changes allowed.

Eventually, the draft becomes an International Standard, the final leg on it's journey. Very minor, or no changes are allowed at this point. Ponder for a moment, the impact that a major change to the RS232 standard would have on business worldwide.

Broadband Communications

What is broadband, the backbone of MAP?

Broadband offers several benefits which, together, make it the logical choice as the factory solution.

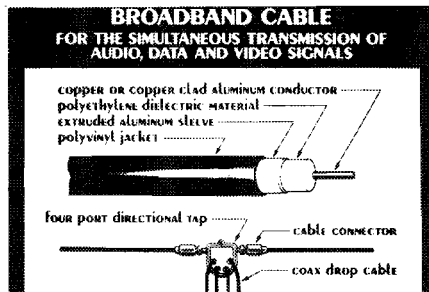
Broadband is the technology used for cable TV today, and for the last 20 years. Due to the requirements of cable TV, broadband components are of industrial strength and immune to heat, cold, humidity, noise and vibration. Broadband is a proven technology under extremely adverse conditions.

Broadband offers multiple channels on a single cable. These channels can be used for data, voice, or video. This allows multi-functionality over only one cable to be strung in the factory.

Broadband offers virtually unlimited distance networks. A single network can span approximately 7 miles, in addition to accessing other networks via bridges.

Broadband was already installed in over 60 GM installations. Most communications within the factory already use broadband technology.

Broadband, 802.4, implies the Token Bus transmission method. Token bus, for reasons pointed out earlier, had been determined to be the most efficient access method for manufacturing environments.



Broadband Components

Now that we understand why broadband was chosen, what are the physical components of a broadband facility?

The four key components of a broadband network are taps, amplifiers, cable and a headend.

The broadband cable itself is strung throughout the plant after a thorough network design and layout. This is in itself, no easy feat. Broadband network design should not be attempted by an individual untrained in broadband data technology. Hewlett-Packard's Network Consultants in conjunction with broadband vendors offer network design services.

Devices, or nodes, are attached to the cable via taps. They look similar to the cable TV attachment on the back of your TV. Taps usually allow multiple devices to attach in one physical place.

Amplifiers are required to boost the signal. Where amplifiers are required in the network design is determined by the combination of multiple factors, such as distance and density of taps.

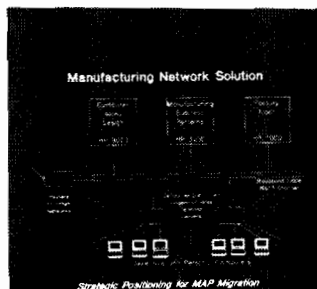
The headend is the brain of the network, providing functionality such as frequency maintenance, token monitoring and regeneration in the event of a lost token.

Maintenance of a broadband network cannot be forgotten during the planning and design phases. The network should be designed with ease of maintenance in mind. Personnel should be designated and trained prior to installation. In the event that a problem should occur on the network, proper test equipment is essential to diagnose and repair the problem.

How Do We Get There From Here?

Now that we understand the basic broadband concepts as well as the benefits of MAP, how do we get there from here?

Let's put the pieces together. The basic reason for installing broadband today is to allow any system to reside logically beside any other system. Another added benefit is that, using terminal servers, any single terminal can access their system of choice instead of remaining dedicated to their host or passing through their host to access another system. This offers the added benefit of reducing terminal and cable costs.



Migration Strategy

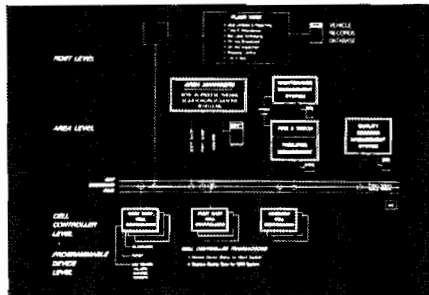
Today, pilots are being implemented. Those pilots are paving the way for products to be released and supported in HP's standard manner. Unless a pilot is part of your future, HP recommends that our customers in a manufacturing environment plan today's automation with MAP in mind.

The first step to MAP is broadband plant installation. This approach maximizes the short term user benefit while minimizing retrofit when MAP is installed, allowing for a graceful migration.

HP's 802.3 LAN product offering has been certified in a broadband environment in a variety of configurations, preserving your Advancenet investment.

The Factory of the Future

The HP Manufacturing Architecture relies on MAP as the communications media within the factory. All devices communicate either through a MAP Gateway or directly attached to the broadband. When customers implement this scenario, Computer Integrated Manufacturing will have truly arrived.



Where are we in this process today?

Today's implementations use MAP Gateways to provide MAP services to devices already installed on the factory floor. A retrofit of proprietary communications services with MAP, if it were available, could not be cost justified. The HP1000 cell controller provides a variety of devices access to the MAP network, relying on our PCIF (Programmable Controller Interface) product.

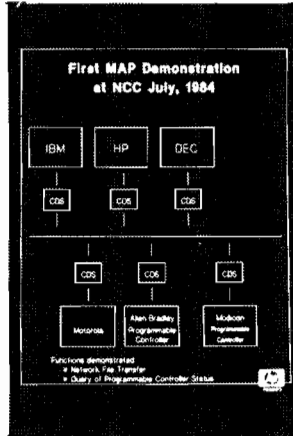
Pilots

Today, through pilots and participation within standards bodies, HP enhances its' leadership position acquired in the MAP arena in 1983.

NCC 1984

In 1984, HP was one of only 6 vendors to support GM's MAP effort by participating in a demonstration of MAP version 1.0 at the National Computer Conference in July 1984. NCC was a landmark in communications history, with MAP gaining recognition as more than a passing whim.

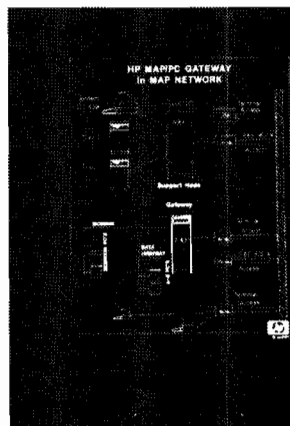
The services demonstrated in 1984, reading files on other nodes and gathering data link statistics, are a small subset of services provided today.



GM/Marion Pilot

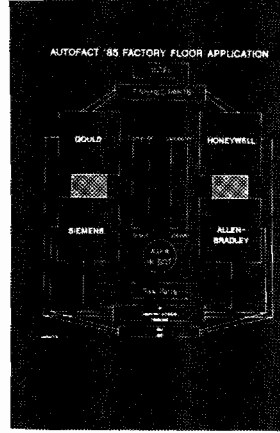
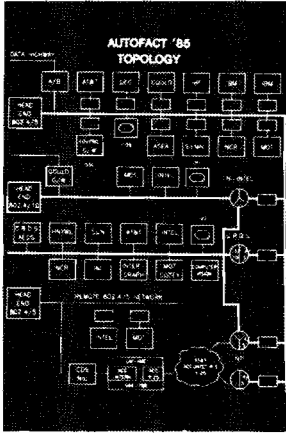
In 1984, HP contracted to provide GM with a MAP version 2.0 pilot for the Marion, Indiana plant. The Marion implementation was destined to be the first cell controller. Installed, tested and functional in July, 1985, the HP pilot was the first MAP pilot to be accepted by GM in the world.

The document cover for "GM MARION PILOT" features the following text: "MAP 2.0", "Programmable Controller Gateways", "Installing to Modcom 584's", "Controlling Process Modules & Diagnosing Functions July 1985". At the bottom, it states "FIRST MAP PILOT TO BE ACCEPTED!" and includes the HP logo.



Autofact 1985

The next test for MAP would be Autofact in November, 1985. In contrast to NCC 84 where only 6 vendors participated, 25 vendors participated in the GM booth at Cobo Hall in Detroit. Autofact was more than a demo. The vendors wrote specific applications that, when combined, formed a small factory, and actually allowed visitors to the quarter acre booth to custom order a toy. The toy was then scheduled (by the HP and Honeywell dispatcher/schedulers) and subsequently manufactured and issued to the visitor.



HP was proud to provide the dispatch/scheduler function between the end user systems taking orders and the manufacturing and assembly lines controlled by various programmable controllers.

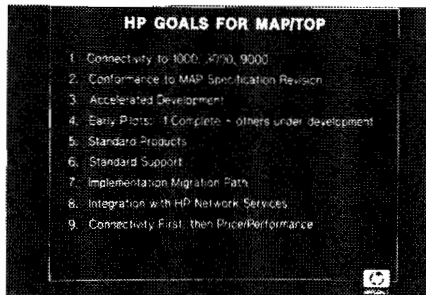
While appearing relatively simple, the Autofact vendors had been testing and certifying their systems to be MAP compatible two shifts, 6 days per week, for 10 months, for only this one application. The MAP 2.1 document had to be modified when problems were found, leading to the Autofact Working Agreement. When we realize that an effort of this magnitude was required to facilitate one application communicating between multiple vendors, the problems to be overcome in a production environment become painfully evident.

Production Pilots

Today, several production pilots are underway, not only at GM but in other Fortune 500 companies as well. The functionality of these pilots varies between companies, but the underlying theme of tying factory communications together remains, with the names changed to protect the innovating. To demonstrate, one would simply change the names of the computers in the Manufacturing Architecture figure.

HP Goals For MAP

Pilots lead to products. HP plans to introduce its first MAP product in late 1986. MAP products are available today to the innovators desiring early pilots. Eventually, we plan to provide connectivity to our entire family of products, integrating existing network services to preserve customers investment. Our co-marketing agreement with Ungermann-Bass paves the way as well as offers a graceful migration. The fluid nature of MAP demands that connectivity be considered first, then performance as we learn more about MAP in true manufacturing environment.



MAP - the CIM Backbone

MAP, as it matures will take its rightful place as a tool, like bisync, RS232 and other predecessors. Computer Integrated Manufacturing is the Factory of the Future, with MAP as its backbone. Without CIM, we cannot compete, and there is no factory in the future. Today, MAP is truly charting the way to the future. HP will see you there.

WIDE AREA NETWORKING - A CASE STUDY

Olivier Helleboid
Hewlett-Packard
19420 Homestead Road
Cupertino, CA 95014

"How networking helped one industry cut paper mountains to bits" (Data Communications).

"Switch to private net satisfies users' need for control" (ComputerWorld).

"Packet switching is firm's link to greater productivity" (Data Communications).

Reading and hearing about such success stories from our industry peers, suppliers or even competitors, are powerful incentives to get started on a similar project. More and more companies are discovering the benefits of wide area networks and are installing their own private data networks.

There is no doubt that the benefits of integrated private data networks are potentially very large. But they should not mask the complexity and investment required for their implementation.

This paper describes a typical implementation of a large company-wide data network from start to finish. Although the leading role is played by a hypothetical manufacturing corporation, most of the process is based on real-live cases. Certainly Hewlett-Packard's experience provided the basis for the presentation, but it is complemented by similar projects at other large manufacturing firms.

The paper's primary goal is to learn from the efforts, mistakes and successes and come out with an even better headline story for Data Communications magazine.

The presentation covers the three major phases of the program: investigation, vendor selection and implementation. Before getting into the network specifics, the first section describes the company and its environment.

COMPANY ENVIRONMENT

The hypothetical firm portrayed in this paper is a multinational manufacturing company. It produces durable hard goods, serving five different markets, industrial and consumer. Annual revenues approach two billion dollars and total employment throughout the world reaches forty thousand people.

The company has a leading position in each market it serves, being one of the top five suppliers in each area. Competition is strong both in the US and overseas. Market share is maintained by constantly increasing the price/performance of its products, but also by strong merchandising programs. The market needs and competitive forces are sufficiently different in the various overseas markets that the company has to put significant localization efforts in the products themselves as well as in its marketing programs.

The company is very decentralized with each of the five business sectors and their respective divisions having strategic autonomy and full profit and loss responsibility for their product line.

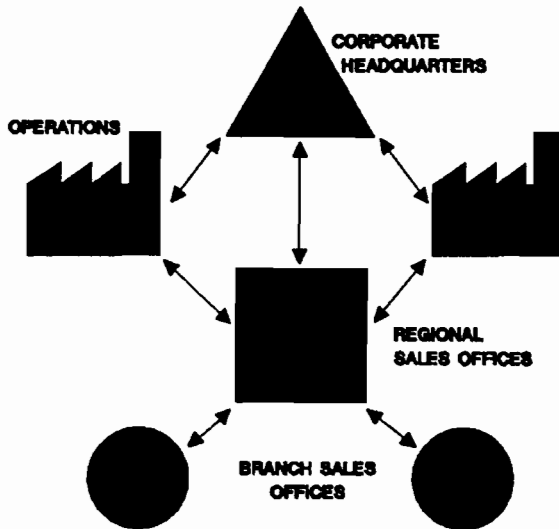


FIGURE 1: COMPANY ORGANIZATION

The organization of the company reflects its decentralized philosophy: a worldwide corporate headquarters in the US housing top management of each business sector, forty worldwide operations, which all have their own research and development, marketing and manufacturing departments, and the separate worldwide sales and service organization with over a hundred sales offices in fifty countries (Figure 1).

Similarly, the MIS and Telecommunications groups are separate and are equally decentralized throughout the company. Each autonomous MIS department in the operations or regional sales offices has decision-making authority over computer and applications purchases for their users. There is a small corporate MIS group, serving the needs of corporate users and establishing computing standards and recommendations for the entire corporation.

Over the last ten years, this decentralized approach had resulted in the establishment and operation of several independent data networks. Each network was optimized for a particular application and was usually incompatible with the other networks. Eight major data networks were identified as being an integral part of the company's day-to-day business:

- sales order processing network (HP equipment)
- sales accounting network (IBM equipment)
- service network (HP equipment)
- operations payroll and accounting network (IBM equipment)
- operations research & development network (DEC equipment)
- batch messaging network (dedicated equipment)

- two electronic mail networks (HP and DEC users).

A couple of years ago, management's attention had started to focus on these separate data networks, primarily from a corporate productivity issue.

Competitive pressures were forcing the company to become more integrated and more market driven. Fast and reliable communications between the sales and operations were becoming crucial to the company's success. In addition, several new products were developed in multiple operations, sometimes located an ocean apart. The project members needed an effective tool for them to exchange information as well as share expensive computer resources.

Cost was not the initial motivating factor behind the data network program, but it rapidly became important. No one knew how much the corporation was spending on data communications: the costs were spread out over the entire organization and they were still relatively insignificant compared to voice communication costs. The first analyses and numbers showed that the company was spending several millions of dollars per year in line charges alone. More alarming was the growth rate of these costs, estimated at thirty to forty per cent per year.

A direct consequence of the initial concerns voiced in upper management was a new objective for corporate MIS: understand the company's remote data communication needs in light of the corporation's business strategy and objectives and prepare a specific recommendation. Timeframe: three months.

INVESTIGATION PHASE - 0 to 3 months -

Three persons were selected to carry out the ambitious investigation within the time limits specified by management: two corporate MIS persons, one from the IBM system operations, the other from the distributed processors (HP and DEC primarily) operations and one person from the corporate telecommunications department. Their common denominator was their several years of experience in the company, in particular in the operations or sales MIS and voice departments.

The first part of the investigation focused on the identification of the user requirements and developing the key attributes of a potential company wide area network.

The team started a comprehensive search throughout the corporation of the existing data networks and how they were used. The list of computer equipment and applications that were involved in remote networking was developed. This part of the investigation rapidly became quite complicated, since it required lengthy and sometimes "politically charged" interviews of the regional sales and operations MIS groups.

Eight major remote computer networks were identified and are listed in the previous section. In addition, the task force found a large variety of "mini" networks often composed of a single leased line linking two computer systems. The team estimated the total data communications cost for the company at two to three million dollars per year.

The task of defining a model for tomorrow's communications was more qualitative, yet more uncertain. The team interviewed dozens of potential network users in a wide range of positions throughout the various groups. The interviews attempted to understand and define communications services that would be valuable to that user in meeting his goals as well as the company business objectives.

Most of the user needs related to immediate problems and focused on three areas:

- connectivity - for example, the sales representatives want to access directly the latest information on pricing, product availability, order and shipment status.

- reliability - for example, the shipping department's daily activity depends on receiving the updated shipment schedule every morning.

- performance - for example, electronic mail users complained about irregular mailing delays, lasting up to 5 days.

After taking into account the inputs from the various MIS departments, the team identified five key requirements for a backbone wide area network:

CONNECTIVITY - any workstation on the network should have the ability to access any information or application located on any remote computer system on the network. Simultaneously with the interactive traffic, the network should allow for the computer systems to exchange batch data. It should be flexible to adapt to the organization's decentralized structure, where peer-to-peer communications are prevalent. The three types of data processing equipment, most commonly found in the company (HP, IBM and DEC), should be able to share the network facilities for their own communications (HP to HP, IBM to IBM,...) as well as for transmitting information between each other (HP to IBM,...).

COST/PERFORMANCE FLEXIBILITY - the network architecture and implementation should offer sufficient modularity and flexibility to enable the company to optimize the cost and performance levels of its network that best suit its needs. In particular, the network should be able to accommodate the increasing needs of personal computer users. Similarly, it should allow for easy integration with future ISDN voice and data networks, when these appear in the 1990s.

RELIABILITY - the company should be able to select various reliability levels that are required for specific applications and that can be economically justified.

SECURITY - the network should provide various degrees of security, in order to satisfy the most stringent controls without affecting the performance and the cost of the non-sensitive applications.

CONTROL - with the wide area network becoming such a crucial element of the company's business, the corporation wants to retain total control of the network. A centralized configuration and diagnostics system is a must for proper network operation. The control system should also provide statistics on network utilization and performance as well as costs, essential for planning and accounting purposes.

The team illustrated these five major qualitative network criteria with two model networks that could be used for cost analyses, one for the regional sales office communications and the other for operations remote data communications.

The final part of the investigation process consisted of an evaluation of the various network alternatives in light of the user needs and network requirements. Four major alternatives were considered: point to point, IBM SNA, public X.25, and private X.25.

Point to point networks were quickly eliminated because of their very limited connectivity. They are proprietary solutions, linking a single vendor's computers. With most of the traffic routing and network management done by the computer systems themselves, a point to point network typically requires more communication lines and computer power to achieve full connectivity. Moreover, the reliability of a point to point network is limited by the availability of the systems on the network.

An SNA network, which was the solution proposed by IBM, did not meet the company's requirements. It did a good job of tying the IBM systems and users together but was not acceptable for the other computer systems and applications. IBM's solution based on gateways and emulation software on the non-IBM machines was very expensive and performance limited. In addition, SNA's centralized network topology resulted in increased communications cost and computer system overhead. The team also realized that the complex SNA solution required a large operations and support staff.

X.25 networks were definitely the best alternative. A single X.25 network could replace all the existing data networks in the company and carry all the corporation's multivendor traffic. It provides total connectivity for computer systems and workstations wherever they are located. An X.25 network is a distributed network with a wide range of transmission facilities and switching equipment. It is designed to fit with the company's decentralized organization and offer the optimal cost/performance alternative.

The team ranked the private X.25 network alternative highest, largely because of its network control benefits, identified as a key requirement for the company. The final recommendation consisted of a private X.25 backbone network with public X.25 network extensions for international connections, access to outside entities and for some particular applications.

The team augmented this qualitative evaluation of the various network alternatives with a financial model. Based on the operations and regional sales office models (Figures 2 and 3), it showed that the communications cost could be reduced by forty to fifty per cent (the monthly line costs for the operations and regional sales office models decreased respectively from \$ 4500 to \$ 2500 and from \$ 9300 to \$ 3700). A company-wide private X.25 network, roughly estimated to cost one to two million dollars, would pay for itself in a maximum of two years.

It was not so much these significant cost savings that influenced management at the review of the investigation results. The user requirements for better communications were felt as being crucial to the future development of the company in the next five to ten years. Some of the investigation findings, bringing

to light inefficiency, poor customer satisfaction and project delays, carried a much bigger cost to the company than any communications savings.

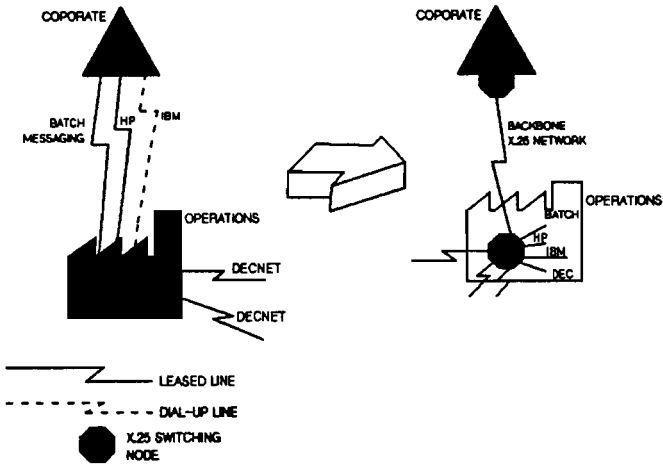


FIGURE 2: OPERATIONS MODEL NETWORK

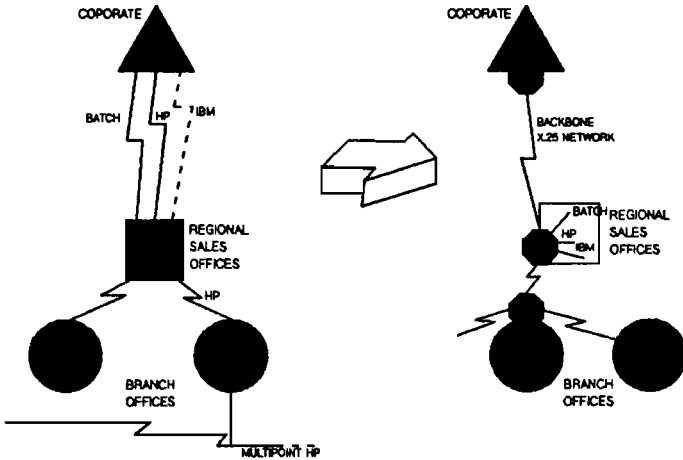


FIGURE 3: REGIONAL SALES OFFICE MODEL NETWORK

Management approved the team's recommendation to implement a company-wide private X.25 network. It was also decided to start using public X.25 networks and X.25 line concentrators in the interim. The team had argued that such an interim

phase would be a very valuable learning experience as well as provide immediate productivity benefits to the company.

The company was now three months into the wide area network project and was ready to enter the second phase: vendor selection.

VENDOR SELECTION PHASE - 3 to 15 months

Early on, the corporation had decided to spend a lot of time and resources for a formal and comprehensive vendor selection. The network would be the backbone of the company's remote data communications for ten years: choosing the right product and the right partner represented a crucial phase of the project.

Simultaneously with the end of the investigation phase, corporate MIS hired an engineer with considerable experience in X.25 networks. The X.25 specialist, in combination with the project leader, would be the principal actor of the request for proposal (RFP) process.

The three month period required to prepare the RFP may seem quite long, but the team was sure that it would translate in greater time and energy savings in the future.

In addition to the general terms and conditions, the RFP contained two major sections: a detailed list of network specifications with their assigned priorities and a simplified traffic model of the company's expected backbone network.

The network specifications were divided in six areas:

- protocol support (X.25, PADs, BSC and SDLC)
- switching nodes (architecture, performance, reliability and redundancy)
- network control center (alarms, statistics, configuration and accounting)
- network security (call screening and auditing)
- gateway to other X.25 networks
- support, documentation and training.

The team felt very strongly about some specific features, listed below, that were not found in many standard product offerings:

- local packet acknowledgement to increase network performance and efficiency
- packet fragmentation and recombination for improved connections to various devices and public networks
- virtual circuit prioritization for a better allocation of the network resources to batch and interactive traffic
- a true X.25 gateway with bidirectional address translation and call filtering
- on-line configuration and modification of the network topology and components.

The RFP presented a simplified but detailed traffic model of the US sales regions and of several US manufacturing sites. The vendors were to propose a network configuration that included location and size of the switching nodes and of the transmission lines, as well as detailed pricing information.

The RFP was sent to eighteen vendors. Seven network manufacturers turned in their proposals two months later.

The list was easily narrowed down to three finalists. That first selection was solely based on technical evaluations and eliminated the vendors who could not match the most important needs, the musts, of the corporation.

For the next selection, the three vendors had to demonstrate their network and provide one test switching node, preferably on the company's location. The node was then tested for compatibility and traffic load factors. The demonstration also allowed the project team to get hands-on experience with the various network control systems.

Simultaneously, reference account visits for each of the three finalists were organized and provided more "live" information.

The tests eliminated one more vendor, primarily for performance reasons.

Until then, most of the selection had focused on product features and performance. The team now carried out an in-depth evaluation of the vendors themselves, including financial stability, company profile, development methodologies, manufacturing processes,...

A business risk analysis played a major role in the final decision, since both finalists had very different profiles. One vendor had an established product, developed and introduced several years before and which many reference sites were using. The second vendor proposed a brand new product, which was clearly superior to the other, but which was less mature and with no real installed base. Eighteen months after the beginning of the vendor selection phase, the company chose the second vendor.

NETWORK IMPLEMENTATION PHASE - 16 months to today

Already fifteen months had gone by since the go-ahead given by upper management. The team had to move rapidly and install the private X.25 network, so that the entire company could reap the benefits from the new network.

In fact, implementation of X.25 networks was already under way throughout the organization, thanks to the efficiency of the new data networking department. Most domestic and international operations and some regional sales offices had connections to public X.25 networks for their urgent interactive traffic. Computer systems and terminals were tied to this network through X.25 line concentrators and X.25 PADs. Dozens of these X.25 devices had been installed in the company. The public X.25 network had certainly brought significant advances in connectivity throughout the company, but was still far from providing a uniform, reliable backbone for all the remote data communications.

Despite the urgency of the situation, the company decided to go through a complete pilot network testing phase that would lead to final network acceptance and production cutover. The pilot network objectives were to fully test the

network, its compatibility with external devices, applications support and its performance and reliability under stress loads and failures.

The pilot network configuration consisted of three switching nodes, the network control system, six PADs linked to terminals and printers, four minicomputers, two X.25 line concentrators and two protocol converters. The equipment was all to be located in a specially built, brand new network control room (Figure 4). A significant part of these first four months was devoted to the construction of this room, ordering and installation of the datacomm lines, modems and of the pilot equipment itself.



FIGURE 4: NETWORK CONTROL ROOM

The other major accomplishment was the establishment of the X.25 network operations group. Ultimately, it would consist of three operators, running the various shifts, and two engineers and two technicians for network planning and support. The engineers and technicians came from other areas within corporate MIS and telecommunications and got immediately started on the pilot testing phase. Plans were made for the hiring and training of the network operators.

The entire X.25 team followed an intensive four-week training program offered by the network vendor. Classes covered network architecture and functionality, installation and maintenance and network operation. Those not familiar with X.25 received a separate three-day seminar on X.25 technology. The department planned to give scaled-down versions of these classes to internal users as the network would be deployed through the organization.

The pilot phase "tested" the relationship with the vendor. Since the product was relatively new, several problems and bugs were uncovered. In most cases, the

vendor reacted rapidly and provided workarounds and temporary solutions until a new product release could be delivered and installed.

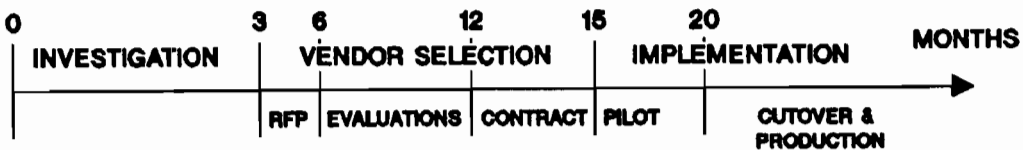
The vendor's support team was not well equipped at the beginning of the testing to deal with such a demanding customer. It quickly turned around and set up a more formal and productive organization, which would benefit all the vendor's customers.

The pilot phase was extended for one month in order to allow a full testing of the new product releases.

In parallel with the pilot phase, the plan for cutover and production of the network was developed. The actual design phase of the production network was limited to the sizing and configuration of the switching nodes and transmission lines. Because of the company's decentralized organization and high connectivity requirements, a switching node would eventually be placed in each operation and regional sales office. The vendor's network design tools defined the characteristics and configuration of each switching node relative to the connectivity, performance, reliability and redundancy requirements.

The cutover plan was based on a very gradual phasing of the existing network to the private network. The first decentralized operations site to install a switching node would run in parallel on both networks for the first three months. Once the second operations site would have been connected to the network for two months, the implementation would accelerate to the rate of one or two switching nodes per month. The implementation plan included the installation of a backup network control system on a separate site.

Deployment of the network overseas presented the additional difficulties of certification. Certification of the switching nodes, which included electrical and RFI safety (UL, VDE, CSA,...), datacomm compatibility (country by country) and finally public X.25 network certification (country by country) had to be planned well in advance. The vendor was responsible for the certification of its products and agreed to a schedule for the various countries with the company. In addition, the vendor had to secure from the US government the necessary export licenses to be able to ship its products to the company's worldwide sites.



Of the twenty months it had taken from the first management discussion to the final network acceptance, six months had been devoted to planning, nine to vendor selection and five for the pilot testing phase. Certainly this company had decided to invest a significant amount of time and resources to the planning and evaluation for its backbone data network. Still, those resources were small compared to the results: a single data network that would, for the next ten years, connect at least twenty thousand users and hundreds of computers in a hundred sites throughout the world.

The company's commitment to the backbone network did not come from the potential communication costs savings. Early on, communications and particularly remote data communications had been perceived as a crucial competitive advantage for the overall corporation. The real benefits of a private X.25 network are improved customer satisfaction with better products and services, as well as increased productivity throughout all the functional areas of the corporation for many years to come.

X.25: WHAT TO DO AFTER THE NETWORK IS IN PLACE

Stephen J. Coya
MCI Digital Information Services
2000 M Street, N.W.
Washington, D.C. 20036

Summary

Many organizations are discovering the advantages of using X.25 packet switching networks to support their data communication requirements. However, it's not until the network is installed that the real "fun" begins: connecting all the hosts and getting them to talk to each other. This is especially true when hosts from different vendors are to be connected to the network. While the OSI model provides a good theoretical framework, many vendors have implemented this model in different ways. As a result, diverse, and sometimes incompatible, protocols must be accommodated and coerced into cooperation before they can communicate.

This paper will review the problems (and solutions) encountered connecting HP3000s, HP1000s, and DEC VAX 11/785s together over a nationwide private X.25 network. It will touch on the physical connections to the packet switch (OSI Level 1), a special Transport Layer protocol that had to be developed (OSI Level 4), and some of the problems encountered with DSN/DS (combination of OSI Levels 3 and 4) and what we had to do to make it all work.

Introduction

We knew from the start it would be interesting. Many companies have made the transition to X.25 networks with relative ease, but this can often be traced to systems where only one vendor host is involved. For example, a network consisting of HP3000s and the selection of a switch vendor to set up the network (sometimes consisting of only two packet switches).

The MCI Mail system, an electronic mail system with hardcopy support, was designed around the use of VAX hosts for the mail application and HP3000s to process and print hardcopy traffic on 2680A Laser printers. The components of this system are connected by a private Bolt Beranek and Newman (BBN) X.25 packet switching network which uses a combination of 9.6 and 56 kbps lines, largely derived from the MCI Telecommunications Transmission System. There are over 50 packet switches in our network. The network interfaces to service hosts and PADs by way of the CCITT X.25 protocol, and utilizes the balanced Link Access Protocol (LAPB) version of CCITT standard High Data Level Link Control protocol (HDLC). Data rates up to 56 kilobits per second are supported for host and PAD access to the PSN.

We drew network (cloud) and system diagrams, examined technical specifications for the hosts, cables, protocols supported and/or required, and the switches, realized what could and could not work, and reached for the bottle of aspirin!

I should mention that there were time constraints involved that did not permit extensive experimentation or research, and the capabilities that exist on today's equipment did not exist three years ago. The system, designed in the latter part of 1982 was built in six months during 1983. Even after details were worked out in joint design meetings, special plugs and cables had to be developed on the fly as we tried to meet our target date, which we did.

Transport Layer (OSI Level 4)

The first "Gotcha" we encountered in the overall design of the system was getting information from the VAX hosts to the HP3000s via the network. All VAX hosts were located in a central facility while the HP3000 based print sites were set up across the country (and eventually in Europe), and each VAX needed to be able to transmit traffic to any of the HP3000s. Tape transfers were out of the question, and the old standby, RJE, did not fit the overall system design. It appeared that HP's DS was the only option available for transferring files to the HP3000s. When DSN/DS was initially developed, it was for point to point communications between HP equipment. It was expanded to support X.25 interfaces, but still required the use of DS as a transport layer protocol. So one still had to use DS to transfer files to an HP3000. Well, DS was never implemented on VAX equipment, and we needed something that would talk straight X.25 with the VAXen and DS with the HP3000s. Enter the HP1000.

The HP1000 was inserted into the network diagrams between the VAX hosts and the HP3000s (we also started using pencils instead of pens to draw the diagrams). We used two LAPB modem cards and two physical connections to the packet network. One modem card was set up to use straight X.25 (DSN/X.25) to communicate with the VAX hosts. The second card was configured to use DS (DSN/1000-IV) to communicate with the HP3000s. Simple? Not quite. While the HP1000 could use DS to communicate with the HP3000, there was no transport layer protocol to communicate with the VAX.

The solution was to design a Simple File Transfer Protocol (SFTP) to accommodate the need for a transport layer. This protocol was implemented on the VAX equipment and on the HP1000s. As its name implies, it is a simple protocol: no CRCs, just simple checksumming and byte counts. It was also implemented on an IBM 4341 which was connected to the network and served the application as the accounting and invoicing host.

So, we had finally identified the host equipment that would be connected to the packet network and had designed a protocol that would provide the OSI level 4 Transport Level between the VAX and HP1000 equipment. Back we go to OSI Level 1, and connect the equipment to the packet switches.

Physical Layer (OSI Level 1)

OSI Level 1 pertains to physical connections, which pins are used for what signals, and the electrical levels of those signals. Unfortunately, there are a number of different standards that have been recommended and implemented: RS232c, RS449/RS422, and V.35 were the ones we had to work with (Please refer to the figure on page 4).

The packet switch nodes (PSNs) supported RS449 physical connections. The switches did not supply a clocking signal, and the hosts were to be connected directly to the PSNs (no modems) at 56 kbps. I mention clocking because at that time neither the HP1000, HP3000, or the VAX could supply clocking at 56 kbps (if they could we didn't know about it). So we had to connect three different hosts that require three different interfaces to a packet switch that supported only one of the three.

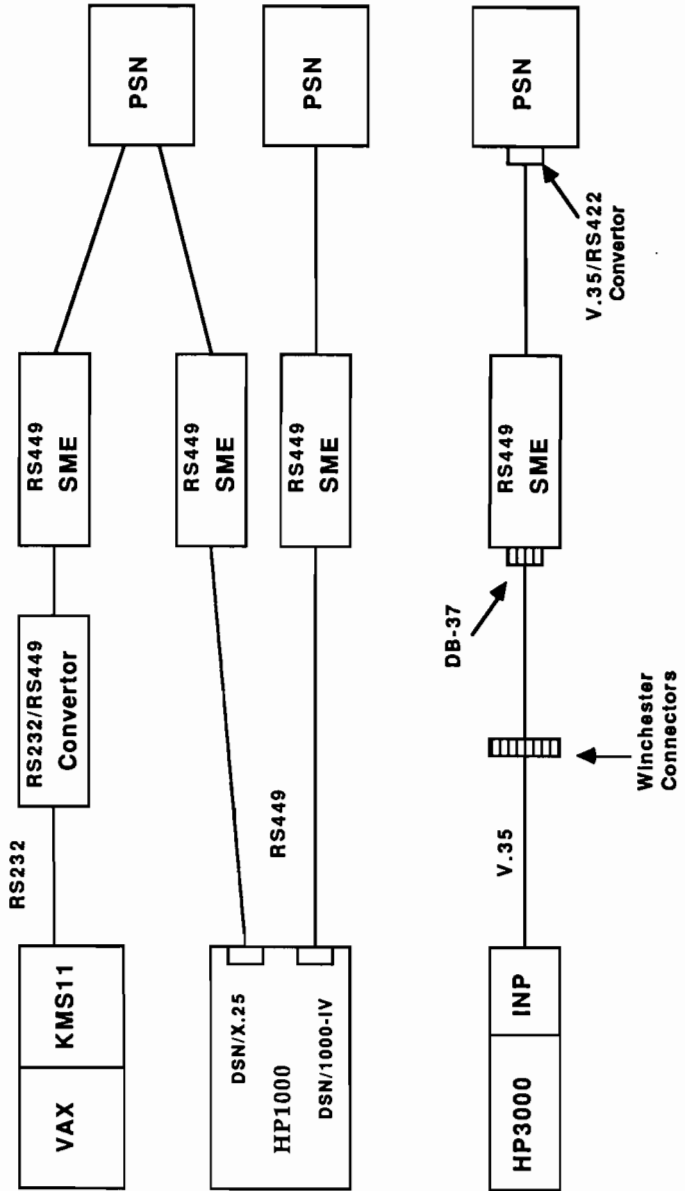
The HP1000 was the easiest as it uses RS449 as a physical interface. All we had to do was insert a RS449 Synchronous Modem Eliminator (SME) between the HP1000 and the PSN to provide clocking at 56 kbps. Since then, we have been successful in operating without an SME as the HP1000 will now supply the clocking signal at 56 kbps.

The HP3000, however, uses V.35 as its physical interface. To establish this connection, a V.35 cable from the INP is used which terminates in a Winchester type connector. The next link is a cable which terminates on one end with a Winchester connector (connected to the HP3000 INP cable), and on the other end with a DB-37 connector. The DB-37 connector is wired to conform to the RS449 specification, the electrical characteristics of which conform to RS422 (RS422 is the electrical specification for signals carried on an RS449 interface). The DB-37 connector is then plugged into a RS449 SME. From the SME comes a cable to a special V.35/RS422 convertor that is plugged directly into the HP3000's port on the PSN. Two hosts down, one to go.

The VAX supports RS232 connections, which is rated at 19.2 kbps up to 75 feet. However, the interface from the VAX to the packet switch goes through a KMS11 processor which has a rated speed of 56 kbps. A shielded cable is used to connect the KMS11 to a RS232/RS449 Protocol Convertor. The next link in the chain is a RS449 SME which is inserted between the protocol convertor and the packet switch.



Physical Connections to the PSN



So we now have all the hosts connected to the network, transport layers are in place, and we can transfer files where they need to go. The system is tested and retested and, eventually, the final bugs are worked out. All done? Not yet - the system must be operated and strange things can happen! I would like to share a couple of "horror" stories with you, problems we encountered over time while operating the system. Since this is INTEREX, I will concentrate on the HP3000, the problems and "characteristics" of the DSN/DS communication protocol.

To Flush or not

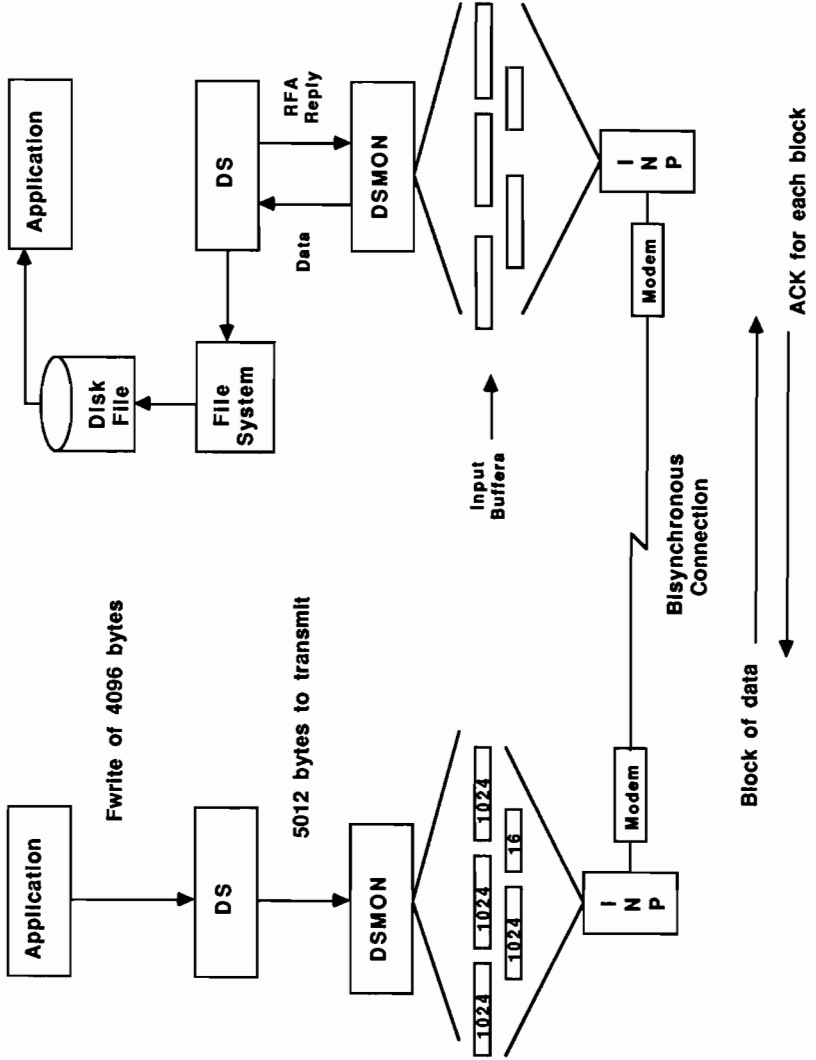
As I stated earlier, DS was originally implemented to support point to point data transfers using bisynchronous communications protocol. Before I describe the "flushing" problem, we should review how the HP3000 handles a DS Remote File Access (RFA) data transfer via a modem link. When an HP3000 is to transfer data to another HP3000, the DS protocol is used to establish the connection to the receiving HP3000 and a Remote File Open command is issued to set up the data transfer. Here's how the data transfer works (Please refer to the diagram on page 6).

Remote File Access

The application program will issue an FWRITE of 4096 bytes. This data is passed to the DS module which attaches the appropriate DS information (headers and trailers). Let's assume this to be 16 bytes of information, so there is a total of 5012 bytes to be transmitted. This information is passed to a module called DSMON which interfaces with the INP. DSMON segments the 5012 bytes of data into buffers with a maximum size of 1024 bytes each. DSMON does not actually send the data to the INP, but passes pointers to the INP, which uses these pointers to transmit the data to the receiving HP3000's INP. So, there are a total of five buffers to be sent to the receiving HP3000 (four buffers of 1024 bytes and one buffer of 16 bytes).

As part of the initial process, DSMON at the receiving HP3000 has set up five data buffers to receive the data from the transmitting HP3000. Using the bisynchronous protocol, a block of data (one of the buffers) is sent to the receiving HP3000. For each block of data transmitted, an acknowledgment must be received by the originating HP3000 before the next block of data can be transmitted. If a positive acknowledgment is received (ACK), the next block is sent. If a negative acknowledgment (NAK) is received, the last block is retransmitted. Each block of data is verified for integrity by computing a CRC checksum and comparing it with the value transmitted. This process continues until all five data blocks have been transmitted and positively acknowledged. At this point, the DSMON buffers at the originating HP3000 are flushed (emptied) and the HP3000 switches from TRANSMIT mode to RECEIVE mode, to await the RFA Reply from the receiving HP3000.

"Normal" Remote File Access via modem link



At the receiving HP3000, when all five blocks of data have been received, DSMON passes the data to the DS module, which communicates with the File System (FS) of the HP3000. This process does the actual write to a disk file and passes the status of the write back to the DS module, which incorporates the status into a RFA reply. This reply is sent back to the originating HP3000 (which is now in RECEIVE mode) and, upon receipt by the application, will switch back to TRANSMIT mode, set up the next FWRITE process, and continue until all the data to be transmitted has been received and stored.

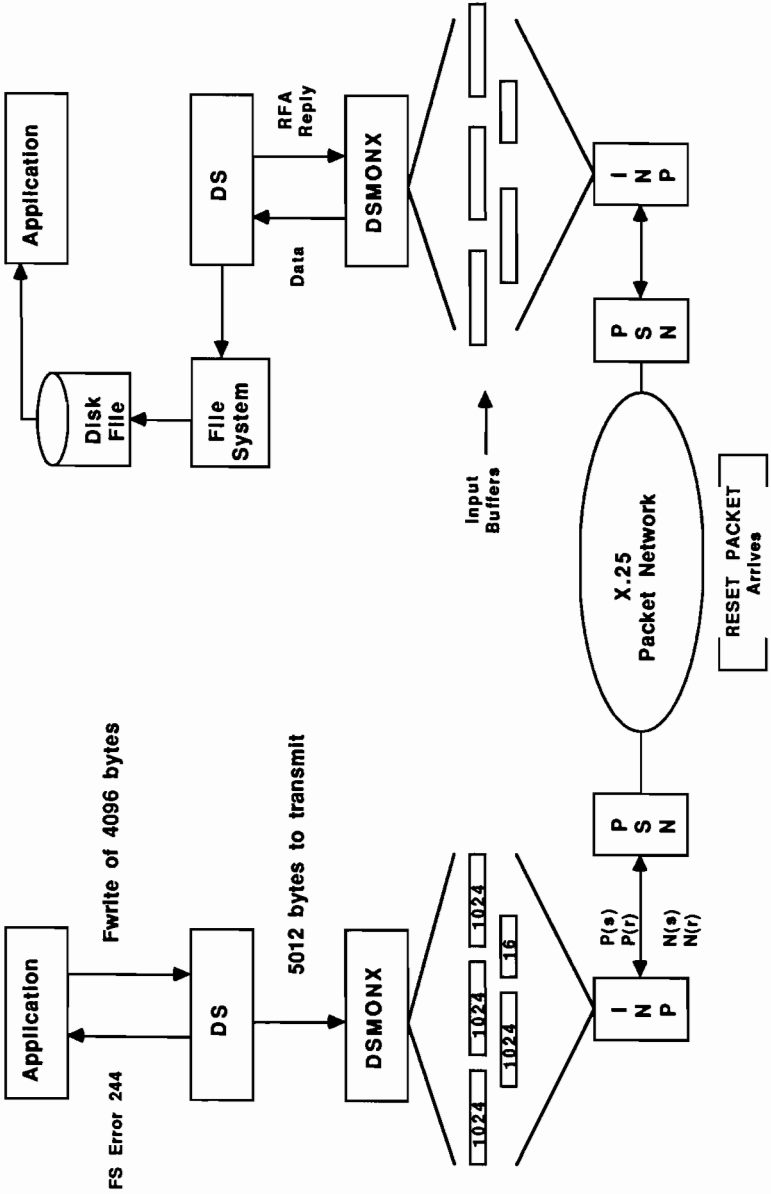
Now enters X.25 based Remote File Access. When HP added X.25 support to DS, they created a second DSMON process called DSMONX, incorporating X.25 protocol but borrowing as much as possible from DSMON's design. Instead of the DS module passing information to the DSMON process, it is passed to DSMONX which will segment the data into five data packets. As we discovered, a little too much of DSMON's protocol design was borrowed.

Consider the differences between two hosts connected by a modem link as opposed to an X.25 network. In the first case, the two INPs communicate directly with each other over the modem. When an acknowledgment is sent, it is from the receiving HP3000 to the originator. With an X.25 network, however, there are at least two (and probably more) packet switches between the two hosts. The packet switch at the originating HP3000 issues acknowledgments upon receipt of each data packet. The packet still must travel through the network to the switch at the receiving HP3000 (Please refer to the diagram on page 8).

MPE IV

As packets travel through the network on their way to the destination PSN (the switch to which the receiving HP3000 is connected), a network problem can result in a RESET packet being generated, essentially stating that something bad has happened and requesting that the packet be retransmitted. This RESET packet is returned through the network to the originating HP3000. If the RESET packet is received by the originating HP3000 before the final packet has been transmitted and acknowledged by the local PSN, the last packet that was transmitted is retransmitted. I'm sure many of you can see the assumptions and the dangerous implications in this response. This design works well in a bisynchronous protocol where each block of data must be acknowledged before the next block is transmitted. But in a network environment, it is dangerous to assume that the last packet was the one which needed to be retransmitted, especially as the level 3 window size increases. But there's more.

Remote File Access over X.25 Network



If the RESET packet is delivered to the HP3000 after the last packet has been transmitted and acknowledged by the local PSN, the buffer has already been flushed and there is no packet to be retransmitted. When this happens, DS just sits there wondering what to do. The application program does not receive any error or indication that something has gone wrong. Recovery from this state required bringing down the DS connection, bringing it back up, and starting the transmission all over again.

The problem was reported to HP, and our SE had to capture a number of dumps to substantiate our claim, especially after HP informed us that 1) they couldn't replicate the problem and 2) the local PSN had acknowledged receipt of the packet, so it was a network problem, not an HP problem! That they could not reproduce it was understandable; they would need to replicate our system switches, configuration, traffic patterns, line speeds, etc. Their second claim was a little hard for us to live with as it wasn't a "network" problem but a SYSTEM problem. HP's position was understandable albeit cavalier, and they did promise to investigate and provide a solution for us.

The solution? You won't believe it. We received a DS patch from HP that essentially said "If network=01 (BBN), don't flush the buffer until the RFA Reply acknowledgment is received from the destination HP3000."

MPE V

Then it was time to move to MPE V. Since it was a new operating system for the HP3000s, and due to the fact that we had a major new release of the packet switching software, we were informed that the special "anti-flush" DS patch would not be on our system when we went to MPE V, and we would have to show that the RESET/flush problem still existed after the upgrade before we could re-request the patch. Well, we upgraded and guess what: the problem remains. However, it appears that we still have the anti-flush patch, but DS changed a little. When a RESET packet is received after the last packet is transferred and acknowledged by the local PSN, the application program is informed that something has gone wrong (an FS error 244 is returned), which terminates the data transfer process. This is a little better in that the application receives an indication that something has gone wrong, but we cannot programatically recover and still have to start the process over. We have also learned a little more about RESET packets and the effect they have on FWRITES.

According to CCITT X.25 Recommendations, RESET packets are sent to the originating host only, not to the receiving host. While the application program on the originating HP3000 knows there's a problem, the receiving HP3000 is still sitting on the line, waiting for the remaining data to fill its buffers so it can be passed from DSMONX to DS, and there is still no way of knowing which packet caused the RESET. There is a need to "beef up" the end to end transport level protocol in DS.

Suggested Solution

There are a number of solutions to this problem, but we realized we would not be too successful requesting HP to modify the basic design assumptions of DSMONX, especially considering the extensive nature of the changes, the amount of time it would take, and the work currently being done on NS/3000. What we need is a way to recover from these situations, and to hell with sophisticated solutions.

In addition to retaining the contents of the data buffer, we have proposed that HP consider adding a "flush and reset" command, either at the DS level or the application level. If at the DS level, the protocol could recover automatically. If at the application level, this command could be sent by an application program when the FS Error 244 is received during an FWRITE operation. When received by an HP3000, it would clear out the current contents of the DSMONX input buffer, reset itself, ACKNOWLEDGE its readiness to the originating HP3000, and wait for the first data packet of the FWRITE. In the meantime, the originating HP3000 would clear out its own DSMONX buffer and, upon receipt of the reset acknowledgment, would start the FWRITE operation.

Our local HP experts in X.25 appreciate our problem and have shown interest in our proposal. We're hopeful, but still waiting.

Error Checking and DS

In my description of Remote File Access communication, I mentioned that error checking in DS is only done at the packet level and not the message level. Again, while this may be appropriate in point to point bisynchronous communication, it does not fit into the packet switching environment very well, as it assumes that nothing can go wrong in the network. We discovered the fallacy of this assumption with RESET packets, but this was not the only problem we encountered.

Lets step back a moment and look at what a packet switch is: a computer. It's a special purpose computer, but a computer none the less. And what makes a computer work? Software. A packet switch contains an operating system, tables, microcode, routing algorithms, etc. Guess what happened when we upgraded the switch hardware and installed a new version of the PSN software designed to run only on the upgraded hardware.

The new equipment and software was tested for some time in a test mini-network we maintain that replicates our operational system, and we verified its functionality, new capabilities, and features. Unfortunately, there's one thing that cannot be tested in our mini-network: LOAD. The new hardware was installed throughout our network and the new software was propagated over some period of time. Two weeks after full propagation we got burned!

There was an obscure bug in the PSN software release that caused an "overlay" of data, overwriting the data contents of a packet in the memory of the PSN. The PSN then calculated the checksum and sent it on through the network. The "bad" data was not noticed by the HP3000 when received, since error checking is done on a packet level and the checksum for the packet was ok (as it was not calculated until after the overlay), and it was accepted. The "clobbered" message was printed and no error was encountered...but there was an error in the message, one that got through the network unnoticed by any of the software!

The PSN software was backed out of our network, the problem identified and corrected, and was re-installed. I mention this here because there IS a deficiency (or hole) with DS and X.25 networks in that these errors are not discovered. What is needed is an end-to-end protocol that can be implemented on both the originator and receiver of the message so that error checking may be done on the message level itself, not just at the packet level which we now see is inadequate. But there is hope.

NS/3000

The latest release of communication software from HP will (eventually) include, among other features, straight X.25 support; decoupling the requirement of DS as the transport layer protocol, though it will still be available and probably widely used. As an alternative to DS, NS/3000 will support the TCP/IP protocol which includes an end-to-end (host-to-host) checksumming algorithm that can be turned on and off. This is message level checking, not packet level, and data errors can be identified and recovered from by the software.

As TCP/IP has been implemented on VAX equipment (The Wollongong Group), it is feasible to use this protocol to permit VAXen and HP3000s to communicate directly with each other without the HP1000 being in the loop. However, there are costs involved that must be considered before we make the decision.

In our application, removing the HP1000s from the communication path would eliminate a central focal point for real time monitoring and controlling of the hardcopy system. On the other side of the coin, it would remove a potential bottleneck as 24 VAXen now communicate with 20 HP3000s through three HP1000s.

The network impact of removing HP1000s must also be considered as this would dramatically alter the flow of data over the packet switched network as all VAXen and HP3000s would communicate over virtual circuits, increasing the traffic and complexity of the network environment. A number of network topology studies would have to be conducted to anticipate the changes to the network, design host redeployment strategies, and determine the network modifications required to accommodate these changes.

Another hurdle is that DEC has not officially "sanctioned" the Wollongong Group implementation of TCP/IP. Indeed, DEC would prefer you using DECNET in an all VAX environment, just as HP would prefer you using DS in an all HP3000 environment. All this really means is that DEC is not actively marketing TCP/IP, and users must ask about it specifically. Just remember, user needs always outweigh vendor desires, especially since it is the user doing the buying.

And finally, we need to wait for HP to release the version of NS/3000 which includes TCP/IP that will run over an X.25 network. TCP/IP is currently available on the HP3000, but will operate over HP's LAN only. Early in 1985, we were told that this would be available by the end of the year. At the Madrid INTEREX conference, I was told it would be available at the end of the year. Next time, I'll ask WHICH year!

Conclusion

In this paper, I have attempted to provide a little insight (and humor) to the difficulties of bringing up a multi-vendor system on a packet switching network; that there is more to do than wait for the network to be installed before you just plug in the hosts. I included a couple of "horror" stories to illustrate that the job isn't done when everything has been connected and initial communication tests completed, but that it is an on-going effort to operate, improve, and even upgrade the system to support current and future requirements.

Because of the audience, I have focused on the HP3000 and DSN problems. However, we have encountered (and solved) network and communication problems with all of our vendor equipment. Being in the telecommunications business, we tend to focus on the problems and quirks of communications support and are always dissatisfied and asking for more. Generally, we are very pleased with the performance and capabilities of the HP3000, especially its ability to be connected to more than one network, though it is too bad the HP3000 does not support network negotiation and is unable to initiate an X.29 call.

MCI believes in open architectures and makes no secret as to how the MCI Mail system is implemented. In fact, a more detailed description of our application and the network was the topic of a paper delivered at the Madrid INTEREX conference last year. I have limited the length of this paper and presentation so that there will be enough time to answer any questions that might be raised, or to discuss other related topics; including the experiences or plans of other organizations that might be building a system from scratch as we did, or who might be planning a transition to a X.25 network based system.

How To Convert From SPL To C Without Making Waves

by

Stan Sieler

Stan Sieler has had seven years of experience on the HP3000, including over four years with Hewlett Packard in the operating system laboratory for the HP3000. He has been professionally involved with programming since 1972, and is currently a member of the Adager R & D team. He holds a BA degree in Computer Science from the University of California, San Diego.

This paper was made possible by Adager, S.A.

Converting SPL to C

0. Introduction

For many years, SPL has been the language of choice for implementing efficient, high level programs on the HP3000.

Before I offend TOO many readers, let me mention that I am primarily referring to programming languages used for major programs that must be fast and maintainable. I realize that every HP3000 language has been used in such projects at various times, and for many people using something other than SPL is the preferred choice (for a variety of reasons, e.g.: not having any programmers who know SPL, or needing COBOL's packed decimal arithmetic).

During the early years of the HP3000, only two reliable languages existed: SPL and FORTRAN. After COBOL II was released, the number grew to three, but SPL remained preeminent. Of the three, SPL was clearly the most tightly coupled to the HP3000. That coupling, plus its ALGOL-like structure, made it the most powerful of the available languages. (That the power could be abused is true, but not relevant here.)

Eventually, other languages started appearing for the HP3000: RPG, BASIC, SPLII, APL, Pascal, FORTRAN 77, Business BASIC, and Fourth Generation Languages. But all of these had various drawbacks, and never managed to displace SPL.

However, the HP3000 is aging, compiler technology is improving, and computer architectures continue to change. As a result of this, many programmers on the HP3000 are looking for a language that will work well today, and will be available on tomorrow's computers as well. SPL clearly works very well today, but its death knell has sounded. HP has made it clear that a native mode SPL will not exist for Spectrum, the successor to the HP3000. What language, then, can we use instead of SPL to achieve the twofold goals of efficiency and portability?

Pascal immediately comes to mind. Unfortunately, Pascal was designed as a teaching language, and lacks a number of features that would qualify it as a replacement for SPL. Within HP, it is not Pascal that is being used to write their next operating system but MODCAL, a MODified version of pasCAL which overcomes most (but NOT all) of the weaknesses of Pascal. But...this language is not available to the HP3000 programmer today.

FORTRAN/77 also comes to mind, but this language has three strikes against it. First, many programmers will look at the name FORTRAN and dismiss it...this isn't fair, but it happens. Second, it is a new compiler on the HP3000 (released in late 1985) and has some reliability problems at present. Third, it isn't clear that FORTRAN/77 compilers will be available on most other machines. This third point, portability, is important for people who might want to move their programs to non-HP computers in the future.

What other languages are available on the HP3000? Until recently, the answer was: none. In 1986, two companys have announced C compilers for the HP3000.

What is C? Can it aspire to be the replacement for SPL?

Converting SPL to C

To quote from the C bible (*'The C Programming Language'* by Kernighan and Ritchie), "C is a general-purpose programming language which features economy of expression, modern control flow and data structures, and a rich set of operators". While I have found the phrase about "economy" to really mean: "we hated to type much, so there is a lot of cryptic stuff in C", I do generally agree with the authors. C provides the programmer with a lot of power (and it is even easier to abuse than SPL!).

C compilers exist for most computers, ranging from the tiny 6502 microprocessor (the Apple II), to the mighty Amdahl 580. Now, C is on the HP3000.

The ease with which the C compiler, and the UNIX operating system, has been ported from the PDP-11 to many different computers says a lot for the portability of C.

The major strike against C on the HP3000 is the compiler reliability question. C is a new language on the HP3000, so we can expect some problems with the compilers initially. However, I think that the compilers will mature quickly. Unlike the FORTRAN/77 compiler, which was probably new code, I expect that both of the C compilers were ported from different machines, which would result in a compiler with less new code to debug. Additionally, I wouldn't be surprised if there is a larger number of programmers who want to use C rather than FORTRAN/77, so the C compilers will be "tested" more quickly.

For this paper I happened to use the C from CCSC. This should NOT be taken as an endorsement for CCSC's C over Tymlab's C, since I currently do not have enough information to make such a decision.

1. Introduction to C

This section is designed to introduce the reader to C, so I had better mention something about coding style in C. As a relative beginner in C, my C coding style still resembles my SPL coding style. I have noticed in the past that when I learn new languages, I often try to impose the style of an older language on the new one. This rarely works to anyone's satisfaction. It is important when using a language to try to code somewhat like the rest of the users of the language, so you can read their code, and vice versa. This shouldn't be taken to extremes...remember that the refrain "but everybody does it" does NOT necessarily make something right. One of the most important aspects of coding is to pick a style and stick with it.

Programs in C consist of one or more functions, including one special function called *main*. C refers to "function" in place of SPL's *procedure*, but the word "function" is actually never used in the language.

An example program that calculates the modulus-11 checksum for a bank account number is shown on the next page, with every line numbered as an aid to referring to them in the text. (The algorithm was extracted from the back of the V/3000 manual.)

Converting SPL to C

```

1. #include stdio.h
2. #include stddef.h
3.
4. short
5.   weights   [6] = { 2, 3, 4, 5, 6, 7 };
6.
7. /*****
8. short calccheck11 (ptr, len);
9.   char *ptr;
10.  short len;
11.  {
12.    short
13.      rslt,
14.      winx;
15.
16.    printf ("MOD11 (%s) --> ", ptr);
17.
18.
19.    winx = 0;
20.    ptr += len - 1;    /*point at units digit*/
21.
22.    rslt = 0;
23.    while (--len >= 0)
24.      {
25.        rslt += (*ptr - '0') * weights[winx];
26.        winx = (winx + 1) % 6;
27.        ptr--;
28.      }
29.
30.    if ( ( rslt = 11 - (rslt % 11) ) == 11)
31.      rslt = 0;
32.    if (rslt == 10)
33.      return (-1);
34.    else
35.      return (rslt);
36.  }
37. /*****
40. main ()
41.  {
42.    char num [10];
43.    short rslt;
44.
45.    while (TRUE) {
46.      printf ("Enter 8 digit number: ");
47.      scanf ("%s", num);
48.      if (num[0] == '/')
49.        terminate ();
50.
51.      rslt = calccheck11 (num, 8);
52.      if (rslt < 0)
53.        printf ("Unable to determine mod11 for %s\n", num);
54.      else
55.        printf ("Rslt = %d\n", rslt);
56.    }
57.  }

```

Converting SPL to C

The outer block of a C program is the function *main*. This one starts at line 40 and declares two local variables: an array of characters called *num* (index from 0 to 9), and a 16 bit integer called *rslt*. Line 41 contains a {, which is C's version of SPL's *begin* (remember...the authors of C hated to type!).

Line 45 is the start of a *while* loop, but notice two differences from SPL here: the boolean expression is parenthesized, and there is no *do* after the boolean expression, just a {.

Notice the uppercase *TRUE* on line 45. Most C compilers differentiate between uppercase and lowercase, unlike SPL. This is unfortunate, as it allows variables like *i* and *I*, which are NOT the same variable.

The traditional C usage is: most things in lowercase, identifiers that are macros (defines) are in uppercase, and when variable names are composed of more than one word, make the first character of each word uppercase (e.g.: *CardCounter*). *TRUE* in line 45 was uppercase because it was a define (found in the file *stdefs.h*). This is the area where I will depart from the existing C conventions. In SPL I use lowercase everywhere, with procedure names being verbs and variable names being nouns. I intend to apply the same rationale to C. Thus, after this paper is finished, I will edit *stdefs.h* to say:

```
#define true 1
instead of:
#define TRUE 1
```

(Note: *TRUE* is defined as 1 in *stdefs.h*...this could cause some problems when interfacing with other HP3000 languages, like SPL, where *true* = -1.)

C has NO I/O defined within it, just like SPL. And, just like SPL (or ALGOL), this was a serious mistake. Luckily, the users of C have developed a relatively standard set of functions to provide I/O capabilities. Most C compilers come with these functions. For example, line 46 will probably compile and run on every C compiler. The standard *printf* function is a lot like the Pascal *write* statement. It provides a variety of formatting capabilities. Line 55 shows two of them: the *%d* in the quoted string is a signal to *printf* that it should grab the parameter after the string (*rslt*), format it as an integer, stick it into the string "Rslt...", and print the result. Since *printf* doesn't do carriage control for you (just as *write* doesn't), the *\n* is a request to *printf* to do a CR/LF at the end of the line. Thus, any *printf* whose formatting string ends with a *\n* is similar to Pascal's *writeln*.

On line 48 note two differences between SPL's *if* and C's *if*. First, there is no *then*. Second, the boolean expression is parenthesized. The lack of a *then* doesn't bother me because I have always maintained that *then* is "syntactic sugar" (something that really didn't have to exist), which is why I always code my SPL *if* statements as:

```
if ... then
...
else
...

```

rather than as:

```
if ...
then      (or then ...)
...
else      (or else ...)
...

```

Converting SPL to C

If you feel you REALLY miss the *then*, and just MUST have it, why, then, C has the answer for you! Just include the following line in your C programs:

```
#define then
```

This is equivalent to SPL's:

```
define then = #;
```

Line 48 shows one of the quirks of C, caused by people who hate to type. SPL, Pascal, and ALGOL use := as the assignment operator, and = as a comparison operator. C uses = as an assignment operator (like FORTRAN and BASIC), and == as the comparison operator. THIS CAN CAUSE A LOT OF MISTAKES FOR SPL PROGRAMMERS MOVING TO C!

Consider the following code fragment:

```
if (a = b)
    terminate ();
```

What this code does is: set *a* to the value of *b*; then, if *a* is non-0 (after the assignment), call *terminate*. The SPL programmer who glanced at the code would have said it meant: if the value of *a* equals the value of *b*, then terminate. Mistakes of this type will happen more frequently if the SPL programmer succumbs to the temptation to use the null define *then* (shown above), because then that statement in C would have looked like:

```
if (a = b) then
    terminate ();
```

Or, in other words, it would have looked so similar to SPL that no mental flags would have been raised saying "look at me closer, I am different"!

Comments in C are different, and similar to Pascal's (* and *). A comment is anything following /* and before the next */. Most C compilers do not allow nested comments, so the best practice is to not use them.

Line 8 shows the declaration of a function, *calcheck11*, which will return a 16 bit integer as its result. Notice that the word "function" is not used. Again, if you feel more comfortable, you could do the following:

```
#define function
```

and then say:

```
short function calcheck11 (ptr,len);
```


Converting SPL to C

C often shows clues to the machine of its origin. It has two interesting operators, pre-increment and post-increment, that were one word instructions on the PDP-11. Line 27 shows an example of post-decrementing. Line 23 shows an example of pre-incrementing...it says: take the value stored in *len*, subtract one, store that number back into *len*, and use it in a comparison against 0. The basic pre/post operators are:

```

++foo    increment foo, use new value in expression.
foo++    increment foo, use prior value in expression.
--foo    decrement foo, use new value in expression.
foo--    decrement foo, use prior value in expression.

```

As we end our very brief introduction to C, let us look at line 20. The += shows that C originated before compilers were smart enough to optimize expressions like:

```
a[i+j] = a[i+j] + k;
```

Instead, C lets the programmer do the optimization:

```
a[i+j] += k;
```

which means the same thing (barring side effects while evaluating *i* and *j*). It's also easier to type.

2. SPL to C Translation

This section discusses techniques for translating SPL to C. It has two major components: the easy part and the hard part.

2.1 Easy Automatic Translation

Much of the conversion of an existing SPL program to C can be very easily automated by your favorite editor. In the following examples I use Robelle's QEDIT, but many other editors would work as well.

Note: the lines below are numbered to facilitate referring to them in the following discussion.

```

1. set shift down 2
2. pq down @
3. c "<<"/**" @
4. c ">>"/**" @
5. c "$include"#include"@
6. c "procedure"(S)" " @
7. c "integer"(S) "short" @
8. c "byte"(S) "char" @
9. c "real"(S) "float" @
10. c "logical"(S) "unsigned short" @
11. c "double"(S) "***TEMP**"@

```

Converting SPL to C

- 12. c "long"(S) "double" @
- 13. c "***TEMP***"long" @
- 14. c ":-"~"@
- 15. c "<="<~"@
- 16. c ">=">~"@
- 17. c "==" @
- 18. c "~=" @
- 19. c "<>!=" @
- 20. c "then"(S) "" @
- 21. c "do"(S) "" "while"(s)
- 22. c "begin"(S) "{" @
- 23. c "end"(S) "}" @

Line 1 tells QEDIT that I will be downshifting SPL style text. Option 2 says: don't change anything within double quotes (").

Line 2 downshifts all text within the source file EXCEPT quoted strings.

Lines 3 and 4 convert SPL comments to C comments. You will have to manually search for SPL comments using the new "throw away the rest of the line" comment character ! (introduced in MPE V/E SPL). With most editors, this might change a few occurrences of "<<" and ">>" that you did not intend to change (for example, if you had "<<" in a quoted string, then you probably did not want it changed to "/*").

Line 5 changes references to SPL \$include files to the C format. Note: be sure to change the include files too!

Line 7 changes SPL 16-bit integers to C 16-bit integers. With CCSC's C, int and short both mean 16-bit integer, but I chose short because I would guess that int might mean 32-bit integers on Spectrum...it is an aspect of C that is NOT well defined.

Lines 8 through 12 change the common SPL variable types to their C equivalents. Note the fancy editing in lines 11, 12 and 13. C uses the terms double and long EXACTLY backwards from SPL. Thus, the simplistic approach of saying: change double to long, and then change long to double would merely change all double and all long to double with no more longs in the file!

NOTE: changes like those in lines 10 and 12 can cause problems if the resulting line is too large for your text file.

Lines 14 through 19 effect the following changes:

| | | | |
|-----|----|----|----|
| SPL | := | <> | = |
| C | = | /= | == |

The extra trick here is to avoid changing things like <= into <== by accident. Again, these change commands could accidentally change items within quoted strings that you did not want changed.

Lines 20 and 21 get rid of the unnecessary thens and dos. Note that line 21 is qualified so that it only affects dos that are on the same line as whiles, thus not touching do/until loops.

Converting SPL to C

2.2 Easy Manual Translation

SPL *defines* can easily be converted to C *#defines*. For example:

```
define twox = (x + x) #, fourx = (4 * x) #;
```

would be changed to:

```
#define twox (x + x)
#define fourx (4 * x)
```

Similarly, *equates* in SPL are easy to handle as *#defines* in C. Another method of handling *equates* might be chosen for some usages. For example, consider an input parser that returns the type of token found, via *equates*, in a variable called *iclass*:

```
equate unknownv = 0,
      tokenv      = 1,
      numberv     = 2,
      stringv     = 3;
integer iclass;    <<always is: unknownv...stringv>>
```

In C this could be changed to use *enum*, which is similar to Pascal's enumerated types:

```
enum scanner {unknownv, tokenv, numberv, stringv};
enum scanner iclass;    NOTE: not all C compilers support enum.
```

2.3 Harder Aspects Of Translation

Some of the conversion of an existing SPL program to C can be difficult. Consider some of the following constructs of SPL:

- address equation at variable declaration;
- subroutines within procedures;
- *move* statement;
- *scan* statement;
- *assemble* statement;
- register usage (*push* and *pop*);
- entry points;
- if expressions.

Some of these constructs will be very difficult to move to C, particularly *assemble*.

Converting SPL to C

Although C does offer a form of address equation with the *union* statement, you may not need to use it since address equation is not used often in high-level SPL programs.

Subroutines pose a large problem. C does not allow functions within functions, which would have been a nice solution. If a short subroutine has call-by-name parameters (or call-by-value ones which are not altered), then it might be converted into a macro. Consider the following SPL example subroutine:

```
integer subroutine min (x, y);
    value x, y;
    integer x, y;
begin
    if x < y then
        min := x
    else
        min := y;
    end <<min sub>>;
```

It can be converted into the following macro:

```
#define min(x,y) ( (x) < (y) ? (x) : (y) )
```

Then, calls to *min* will work as before.

Sometimes, a subroutine can be converted into a separate function. Ideal candidates for this are subroutines that use none of the local variables of the surrounding procedure.

If a subroutine only uses a small number of a procedure's local variables, then you might consider making it a separate function and pass those variables in as additional parameters.

The SPL *move* statement can be translated into a function call:

```
move p1 := p2, (10);
```

translates into:

```
move (p1, p2, 10);
```

This, of course, assumes that the original SPL *2move* was a byte-oriented, not word-oriented. Note that you may have to write a *move* function yourself in C. Alternatively, the following macro accomplishes the same SPL move:

```
#define move(p1, p2, len) {short ktr; for (ktr=0; ktr < len; ktr++) \
    p1[ktr] = p2[ktr]; }
```

(The backslash (\) indicates that the line is continued on the next input line.)

Or, as another alternative, some C compilers provide the routine *memcpy* for moving bytes:

```
#define move memcpy
```

The SPL *scan* statement can be handled in a similar manner.

There is almost no hope for the SPL *assemble* statement. But, luckily, there is almost no reason for it to be in your SPL programs anyway.

Converting SPL to C

Similarly, any SPL code that explicitly accesses any hardware registers (e.g.: *Q*, *DB*, *X*, *S*, *DL*) will have to be examined by hand.

Some C compilers for the HP3000 do not support entry points. While most programmers do not use entry points, and therefore will not miss them, some do. For example, many HP programs in PUB.SYS have documented entry points (e.g.: EDITOR) that drastically affect their behavior. I have a policy of having a HELP entry point in every program I write, so that a new user only needs to say:

```
run tapedir, help
```

to get safe help information (such programs ALWAYS terminate after delivering the help information).

The C *if* statement differs in an important way from the SPL *if* (besides in appearance). In SPL, the *then* part will be executed if the boolean expression is true...but "true" is defined as "bottom bit is a 1". If C, the *then* part will be executed if the expression is non-0. This difference can drastically affect programs. Consider the following SPL code that checks to see if a variable is an odd integer:

```
if logical(k) then
```

If this were translated to C without thinking as:

```
if (k)
```

then it means: "if k is not 0". Instead, a proper C translation would be:

```
if (k & 1)
```

which does a "bitwise and" with *k* and 1, returning 0 if the value was even and 1 if the value was odd.

3. Conclusion

C is a very powerful language, and exists on many different computers. Because of this, I feel that it is an excellent candidate for replacing SPL.

I am not abandoning SPL right now, but I am going to start coding SPL with the thought in the back of my mind that someday, soon, I will be moving to C. This knowledge will provide an incentive to avoid those features of SPL that ARE hard to convert to C (or to ANY other language). Additionally, I plan to work further in C on both the HP3000 and on various microcomputers in an effort to get "up to speed" in it as fast as I can.

Remember...keep an eye on C...it may be the language in your future!

Commercial Spectrum Progress Report

by: Simon, Rick

**We regret that this paper
was not received for
inclusion in these proceedings.**

Meeting the Challenge: An Inside Look at the Spectrum Testing Program

Donald M. Barnett
Hewlett-Packard Company
19447 Pruneridge Avenue
Cupertino, CA 95014-9974

Wayne E. Holt
Software Research Northwest, Inc.
Route 1, Box 610
Vashon, WA 98070

Introduction

The Spectrum Project is the single largest undertaking ever attempted by the Hewlett-Packard Company, involving dozens of divisions and thousands of engineers. The requirements for essential testing and quality assurance demand an effort that rivals that of the actual project.

This paper will define the scope of the commercial Spectrum program in terms of the diversity of products that must be integrated and tested. The testing process will be described, its goals will be examined, and the level of effort necessary to accomplish those goals will be explained.

The focus of this paper will be on the process used to measure the reliability of the core system products produced by the project. The authors are involved with the reliability stage of testing, one of the final steps prior to shipment of the products. The process, methodology, and actual mechanics that are involved with volume, stress, and reliability testing will be discussed.

The Spectrum Program

Spectrum is not a product; it is a program of projects that have as a goal the creation of products that meet certain design criteria and specifications. As long ago as 1979, Hewlett-Packard began work on a number of projects to prove out the feasibility of an advanced version of the MPE operating system, one that would take advantage of the advances in technology that had occurred in the mid-to-late 1970's and those that were anticipated in the 1980's and beyond. In 1981, this resulted in the decision to proceed with a major project to create such a product, and that project was called HPE. In 1982, the decision was made to proceed with the engineering of a new hardware architecture that would include RISC concepts, and that project was named Spectrum. In 1983, the decision was made to combine the new version of the operating system with the new hardware and the commercial Spectrum program, the next generation HP3000, was born.

The Spectrum program has been designed to bring the various product families (HP1000, HP3000, and HP9000) together in a single hardware architecture, while permitting different operating systems to be built for use by the customer as the application dictates. Specifically, the two operating systems that were targeted at the inception of the program were HP-UX

and MPE. This paper's focus is MPE and the resulting requirements for the commercial program.

The MPE/XL operating system is a new product. It has the unenviable task of behaving precisely like its predecessor, MPE, while running code which has not been recompiled, and supporting a whole new set of capabilities as well. If the code is recompiled, it must deliver vastly increased performance, using techniques radically different than those of its predecessor. Further, it must support a wide range of products already in place on the HP3000, plus several new ones designed expressly for the new architecture. Some examples of the product diversity are listed below:

- o The File System, the Command Interpreter, Editor, Sort, FCOPY, VPlus
- o Pascal, COBOL, FORTRAN, Toolset
- o KSAM, Turbo IMAGE, HP-SQL, HPIMAGE, Dictionary
- o Rapid, TDP, etc.

The level of effort to integrate this software, test the engineering quality, and deliver the product to the user is enormous. Over 140 engineers are involved in the test and integration effort alone; in addition, each Lab, Division, and Operation that is involved also has substantial testing programs in place. Figure 1 illustrates the product flow from each development lab through the final testing process. The efforts of each lab must be centrally monitored to insure orderly completion of milestone objectives.

Quality is the primary objective of the Hewlett-Packard testing program; to achieve the highest levels of quality, it is essential that those people involved in the development process are aware of the criteria by which the product will be measured. The Software Engineering Lab (SEL) is part of the Information Technology Group (ITG), the group that is primarily responsible for the quality measurement of the product that is delivered to the customer. SEL has adopted the following mission statement to guide the collective efforts of the engineers assigned to the test and integration process:

Software Engineering Mission

To assure that each Spectrum Program product achieves the optimal balance of highest quality, most effective support, and minimum time to market, which maximizes HP's success in the marketplace, and to champion throughout the corporation concepts and technologies to achieve these goals.

The structure of the testing process, and the measurement of how goals are achieved, are based on this mission statement.

The Philosophy of the Testing Process

ITG management clearly identified the creation of an advanced testing laboratory as an early and important goal of the Spectrum program. In turn, the SEL management sought to identify the major tasks that would confront the development teams:

The problem: how do you ensure the quality of a totally new product consisting of millions of lines of source code on a radically advanced architecture and still fulfill the requirements of both the customer and the company?

The solution: test the product **earlier, deeper, wider, in parallel, incrementally**, and with a **goal**. The SEL organization developed this approach to the problem as a result of extensive research conducted during the formative stages of the Spectrum program.

Traditionally, many software projects have done testing as an afterthought to basic research and engineering. By testing **earlier** in the development process, defects which would eventually plague the lab engineer (and ultimately the customer) are detected before they become critical, and a higher quality product is produced.

The concepts of depth (**deeper**) and breadth (**wider**) were developed as a result of an historical analysis of earlier software testing projects at HP. It was found that, while testing in the past was functionally complete at a command level, the coverage of possible combinations (depth) and coverage of possible paths (breadth) was limited.

To further increase the coverage of the testing effort, **parallel** approaches to product testing were deliberately planned. These efforts were orchestrated such that the testing methodology of any given testing team served as counterpoint to the approaches taken by other teams examining the same software modules.

Any ambitious project will tend to overwhelm the engineers involved with elaborate detail that is constantly changing. To manage this problem, the number of variables allowed in any given testing process were tightly controlled so as to facilitate faster and more accurate problem resolution. The **incremental** testing concept allows only limited changes to pass through the testing process in a normal cycle.

By far the most important concept in the Spectrum testing program is the identification of and agreement to a common set of **goals**. Specific detail criteria have been developed for each stage of the testing process. These criteria enumerate **goal** levels for such items as defect densities, code coverage, load densities, depth, breadth and performance. Not only must a software module execute "as designed" in the testing environment, it must do so *while also meeting these additional quality goals*.

These **goals**, set at strategic points throughout the program, have been primary factors in motivating the engineering and technical staffs to develop the highest quality computer products ever produced by Hewlett-Packard.

The Structure of the Testing Process

It is difficult to define the structure of the Spectrum testing process without first defining a few terms:

Integration. The process of bringing various software modules together into one cohesive product, forming an integrated unit.

Unit Level Testing. Testing done by the development engineers and lab testing teams to certify software before submitting it for system level integration.

Regression Testing. The process of re-running a specified suite of tests which have passed previous test processes.

System Testing. The process of running specified sets of tests against a fully integrated system.

Figure 1 illustrates the hierarchical structure of the commercial Spectrum testing process as it is currently implemented. In this process, the development engineers submit their individually tested modules to a lab software integration team (LSIT) which builds an executable subsystem. The LSIT then returns the integrated software to the submitting engineers as well as to a lab software test team (LSTT). The LSTT executes a series of unit level tests designed to certify that the individual modules submitted to the LSIT work together as a unit. When the LSTT has given its blessing to the software, it is submitted to the systems-level integration team (SLIT). The SLIT builds an appropriate software release that includes all of the products submitted, and distributes the software back to the development lab for regression testing with the unit level tests. In parallel with the development lab's regression testing process, the software is also given to the systems level test teams (SLTT) to begin system testing.

System testing is divided into the following groups:

Diagnostic testing allows an engineer to do virtually anything to a system in an attempt to cause it to fail. This type of testing is also referred to as destructive testing, and is typical of the kinds of activities that are often associated with maniacal hackers.

Security testing is chartered with the task of ensuring that the accounting structure and security restrictions on MPE/XL are operationally identical to those currently in place on MPE.

Recovery testing is responsible for two areas. First, the system powerfail software must properly handle an orderly shutdown of the system when a power outage occurs. Second, the recovery software for both the operating system and the data files must correctly restore the system state to where it was prior to the shutdown.

Configuration testing is charged with the task of ensuring that various hardware and software combinations are operationally correct.

Operator Certification testing is responsible for ensuring that interfaces between the operator and the machine are functionally correct, consistent, and reliable.

Migration testing validates a wide variety of transitional paths for applications from the MPE environment to MPE/XL to ensure minimal impact to the customer.

Performance testing measures the system throughput capacity and identifies areas within the system where optimal performance is not present.

Volume/Stress/Reliability testing measures the capacity of a system to remain operationally sound when subjected to loads of varying sizes and content.

When viewed in perspective, the Diagnostic, Security, Recovery, Configuration, Operator Certification, and Migration testing groups have focused objectives, while the Performance and Volume/Stress/Reliability testing groups take a broad view of the system.

Volume and Stress testing may be viewed as subsets of the more rigorous Reliability testing. Volume testing is defined as the process of

- (1) identifying specific limitations within the system, and
- (2) testing to the identified limits of the system, and beyond.

A Volume test is considered successful if no errors occur when testing the outer limits of the system, and when exceeding those limits, the errors generated are appropriate to the situation and the system recovers gracefully without loss or corruption of data. Volume testing is normally done in a standalone environment. An example of a Volume test would be opening the maximum number of files allowed for any given process. After validating that the files that were intended to be opened were actually opened, the test would then attempt to exceed that limit. If the system reacted appropriately, the files that were originally opened would remain open, the data within the files would remain intact, and the error message returned to the terminal would be accurate.



Stress testing is defined to be the orchestrated loading of the overall system while one specific area is driven to peak demands. In a simplified example assume a system capable of the following four functions: file management, task management, terminal communications and data communications. If the test target is file management, then the following events would occur:

- (1) The other three areas of the system would be loaded to a level which collectively utilizes approximately 40 percent of the system's resources.
- (2) The target area tests would be arranged so as to utilize as many of the file management system resources and capabilities as possible.

Unlike Volume testing, Stress is not intended to exceed specific limitations of a given target but rather approach the many individual limits within a target area.

Reliability testing, while appearing to be an extension of Stress testing, has several key differences. Where Stress testing is focused on given targets, Reliability testing is much more random as to which areas of the system will be exercised. Also, Stress testing generally runs for relatively short periods of time while Reliability testing must run for extended lengths of time with varying and diverse loads. Often a Reliability test will appear to be stopped due to lack of activity. Periods of sleep are desirable and often intentionally programmed.

One of the primary goals of reliability testing is to execute as many combinations of events in a given period of time as possible. The method employed to accomplish that goal involves the automation of tests using a terminal session simulator, which is capable of randomizing test selection and execution while still maintaining the ability to repeat a sequence of events. By developing such a tool, the test team is able to simulate long-term typical customer system usage within a short period of time.

Volume, Stress and Reliability testing are an incremental process that is increasingly complex with each successive step. Figure 2 illustrates the migration of the process from standalone Volume tests, through the Stress test phase, and finally the Reliability phase. By following such a controlled process, the Volume/Stress/Reliability testing program is able to reveal defects in the products that otherwise might only have appeared at the customer's site.

Conclusion

The testing process that has been discussed in this paper is an integral part of a larger quality program instituted by ITG. As noted in the *HP Journal*, "... that program is designed to ensure that the Spectrum software is the highest quality software Hewlett-Packard has ever shipped by using pragmatic measures to define software quality and evaluating progress toward established goals. HP will continue to improve and refine its software development practices to provide software of *increasing quality and decreasing cost* to its customers."

This paper has presented a synopsis of the testing process within the commercial Spectrum program. An oral presentation of the material will cover the material at a detail level not appropriate to a written presentation.

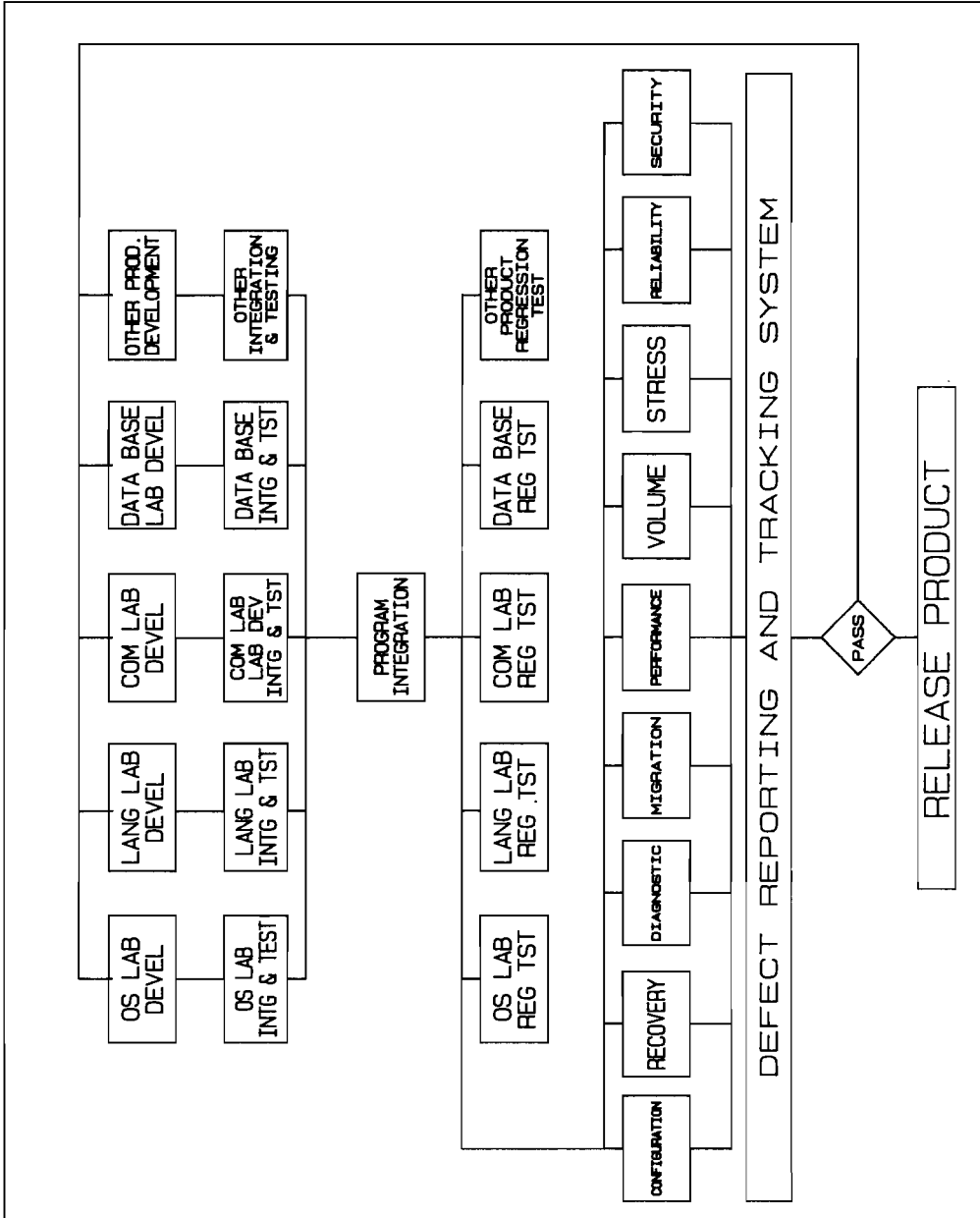


FIGURE 1

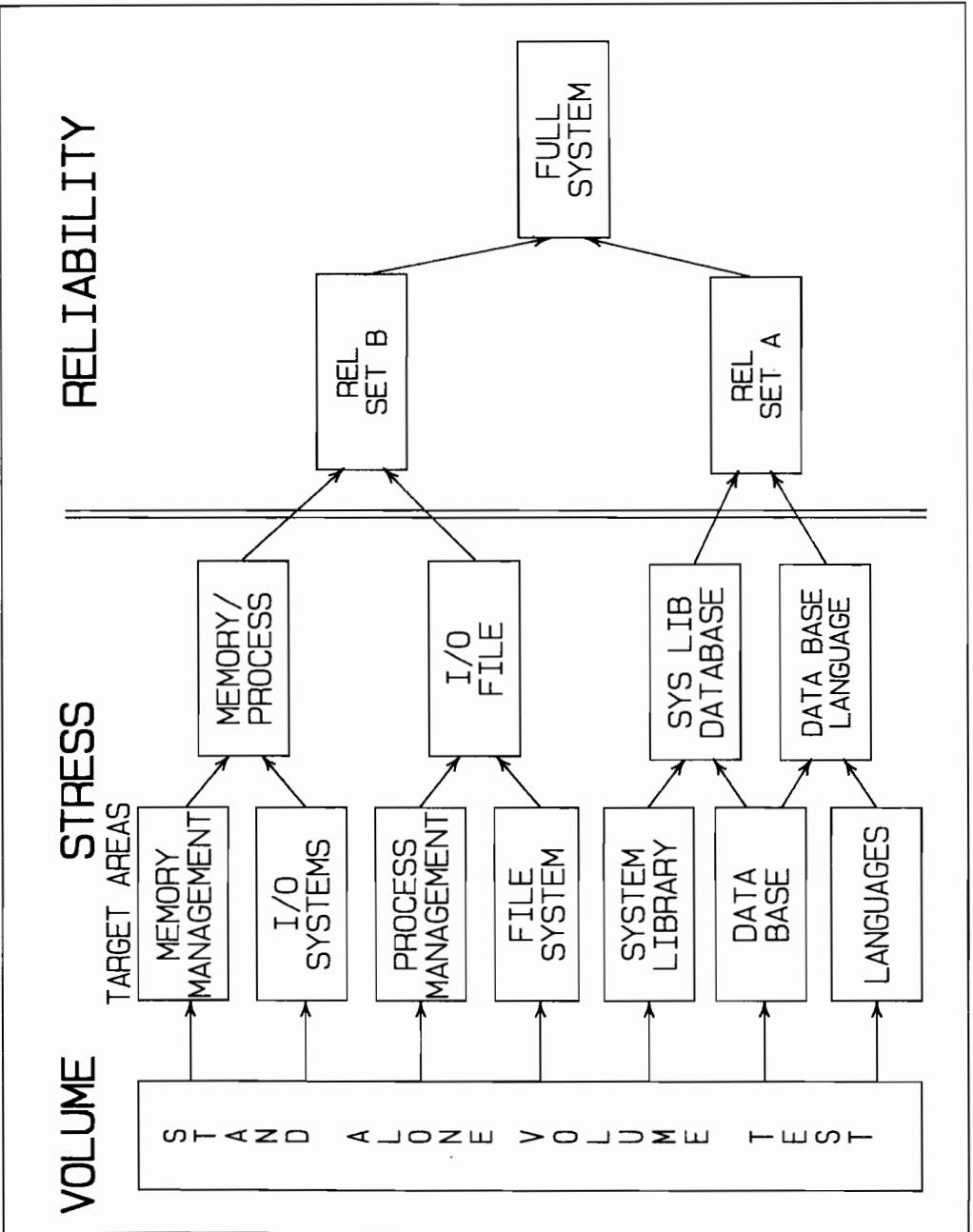


FIGURE 2

Migrating to the series 900's — Variables Affect. Sys. Perf.

**by: Friedrich, Richard
McBride, Becky**

We regret that this paper
was not received for
inclusion in these proceedings.

Migration Solutions for MPE/XL

by: Cargnoni, Lawrence J.

We regret that this paper
was not received for
inclusion in these proceedings.

MPE XL Organization and Direction

**by: Courtney, Lee
Garcia, I. Janet**

We regret that this paper
was not received for
inclusion in these proceedings.

Programming the New Generation of Digital Computers

A R.I.S.C. Tutorial

by Jeff Hecker
Arens Applied Electromagnetics
P.O. Box 329
Gaithersburg, MD 20877

After all of the gossip, rumors, guesses and second guesses, Hewlett-Packard has announced the first products to come from the Spectrum project. Despite all of the claims of compatibility, many people remain unconvinced. How can a completely new machine really be compatible? How can a completely new machine with a completely new instruction set be able to execute old programs? How can programmers take advantage of the new instruction set for highest possible performance?

The new HP computers follow a recent trend in the computer industry towards what are known as Reduced Instruction Set Computers (RISC). RISC computers in one sense, might be considered to be a throwback to the very first days of programmable computers. Then, there were only a very few instructions such as load, store, add and compliment. Not even a subtract instruction! To subtract two numbers, the programmer would fetch the first operand; then fetch the second operand; compliment the second operand; add it to the first operand; and finally store the result back into memory.

In this case, a simple operation took twice as many instructions as it would first seem to require. Also these were the days of machine level and symbolic assembler programming. There was no such thing as a FORTRAN or COBOL compiler. It became clear to computer designers that it would be much more efficient if the computer could perform more complex operations. This way, the number of instructions coded by the programmer could be reduced. Fewer instructions to be coded translates directly to lower costs and to higher software reliability. (Sound familiar?) The current generation of Complex Instruction Set Computers (CISC) was born.

Over the years, instruction sets continued to expand. Not only could computers subtract, but they could multiply and divide too. Sophisticated operations such as memory to memory copy, floating point math, iterative loop instructions, and even memory block checksum calculations were incorporated into the computer's instruction set. When microprocessors were introduced in the early 1970's, one measure of sophistication was the number of different instructions they could execute.

Over those same years though, software technology was improving as well as hardware technology. FORTRAN followed by COBOL, PL/I, Pascal, C, Ada and others have been developed to further improve the efficiency of the programming task. Programmers no longer had to worry about which particular machine instruction was being used. In fact, today's programmers are so thoroughly insulated from the actual computer which executes their programs, that one actually thought that a FORTRAN formatted WRITE statement was actually carried out by hardware.

High level languages have caused two sweeping changes in the computer industry. First is program portability. Since a programmer (and therefore his program) are no longer concerned with the actual computer instructions, he can take his program to a completely different computer with (in theory) no changes.

The second change was much more subtle. With the use of high level language compilers, a very few people decide which of the computer's instructions actually are ever used. When a compiler is written for a particular computer's instruction set, it must translate the language source statements into the appropriate sequence of machine instructions which will accomplish the needed function. Once the compiler is written, only those instruction sequences generated by the compiler will ever be executed.

Over the years, compiler writing has advanced as well. Compiler writers no longer just arbitrarily chose instructions from the CPU's instruction set. Execution speed was also being considered. The classic example is the case of the VAX loop instructions. They are expansions on the standard "Decrement and branch on not zero" instructions found in many CPUs. But they executed so slowly, that compiler writers issued more traditional sequences of simple instructions which executed faster. This instruction (and others like it on many different computers) are the excess baggage that CISC computers must carry around with them when a new processor is designed. Many of a CPU's instructions may be rarely, if ever, executed. This is particularly true when one group designs the hardware, and an independent group develops the software and compilers.

This phenomenon has not gone unnoticed by the computer designers. Beginning in the late 1970s, several researchers began to independently measure which instructions were actually being executed by CISC machines. Their results were another variation of the 80-20 rule. 80 percent of the time, computers were executing the same 20 percent of their instructions.

At the same time, another phenomenon was beginning in the computer industry: the UNIX operating system. UNIX and its native programming language "C", were appearing on more and more computer systems every day. Since all of these UNIX systems came from one source, they all used precisely the same C compilers. Suddenly there were hundreds of machines executing the same 20% of their instructions 80% of the time.

These developments triggered two reactions from computer designers. One was the C-machine. The C-machine was designed specifically to execute C programs. Its instruction set was taken from those actual instructions generated by the UNIX C compiler.

Secondly, academic researchers considered the idea of eliminating the rarely used instructions from the computer. Taking the results from their measurements, they determined which

instructions were actually useful, and which were not. Many of the instructions which were relatively unused are also the same complex instructions which have been added over the years. RISC machines were born.

RISC machines offered great potential benefits to computer circuit designers. Eliminating some (or all) of the more complex instructions allowed circuit designers to build a faster system for an equivalent cost. Fewer different instructions also require fewer different CPU components. Fewer CPU components require fewer transistors to implement them. Fewer transistors require less silicon per chip produced. Less silicon per chip results in a higher yield. Fewer transistors also require less power. Lower power increases reliability. There is almost no fundamental reason not to implement a RISC CPU.

Recently, several computer manufacturers have introduced RISC CPU based systems to the market. The Bolt, Berenek & Newman (BBN) C-70 is a C-machine, designed specifically to run the UNIX operating system. Pyramid's 9X systems are VAX-class minis designed for the general market. Several other RISC machines have also been announced and delivered.

In Hewlett-Packard's case, simply implementing a new RISC based computer system would have alienated a large potential customer base. Not only must the new generation of HP RISC based systems have more power than the current HP 1000/3000/9000 systems, but they must be compatible with these systems as well.

There are several different approaches to compatibility. The simplest is to require each application program to be re-compiled on the new system. Another method is to actually include parts of the old machine in the new machine. The third approach is to emulate the old machine with the new machine.

The first method is in common practice in the UNIX community. Compatibility is defined at the source code level. It is not unusual to require the complete re-compilation of all programs even when an O.S. update is installed. This is particularly likely on some UNIX look-alike systems such as Microsoft XENIX. Fortunately, UNIX provides many more flexible and automatic program generation tools than any other operating system, making complete re-compilation of any program very simple.

The second approach was used by Digital Equipment Corp. when their new 32-bit computer system, the "VAX" was first developed in 1978. At the time, the PDP-11 was widely installed, and DEC had the same problem which HP has had to face. DEC's solution was to include a PDP-11 processor as part of the VAX processor. DEC customers could run their existing PDP-11 programs on the VAX unchanged, until the programs were converted.

The approach chosen by HP is to emulate the HP3000 instructions in software. A program will examine each HP3000 instruction, and branch to the appropriate subroutine (dubbed "milli-code" by HP) for execution.

Of the three possibilities, the first seems least expensive, fastest to implement, achieves the highest performance, but requires the most effort by customers making a conversion. Also, each customer must have the source code for each of their programs. This is not possible when customers buy their software from companies such as HP. The software developer is,

therefore, required to convert all of the software before the system can be delivered, yielding a false economy.

Including an HP3000 processor in the new machines would be contrary to the entire RISC philosophy. The cost of the additional circuitry, and additional power consumption would be passed along in each unit, even after conversion to the new system. However, this method is most reliable in the sense that there are no new compilers or interpreters which might not function properly in every instance. This is why DEC chose this path.

An interpreter is HP's compromise between the hardware integrity and having to convert the entire HP applications software catalog. An interpreter can also execute a program for which the source code has been lost, or a program which has been patched for a particular configuration. The other side of the coin is the severe performance penalty of interpreted operation.

How do instruction set emulators work? How can one machine execute another machine's instructions? Why is there a performance penalty, and how severe is it?

The new HP3000 models from the Spectrum project will be able to execute existing HP3000 programs in "compatibility mode". Since the new computer system uses a new instruction set, the CPU itself will not be able to execute the instructions. A system utility will be invoked transparently to the user in order to execute the program in HP3000 compatibility mode. This utility is the emulator or interpreter.

The interpreter is a Spectrum native mode program which uses the HP3000 program as input data. Consider the BASIC/3000 interpreter currently available. BASIC is an HP3000 "native mode" program which uses a BASIC program as input data for execution. The principle is precisely the same. The HP3000 interpreter is a Spectrum native mode program which will examine each HP3000 instruction to be executed, and then execute it.

Interpreters, by their very nature, incur a tremendous amount of overhead. But how much? Consider the steps which must be carried out by the interpreter for each emulated instruction:

- 1) Check that the instruction pointer does not point beyond the current code segment limits.
- 2) Fetch the instruction to be emulated from memory.
- 3) Increment the program instruction pointer so that the next instruction can be fetched from the correct memory location.
- 4) Determine which instruction this one is, either by look-up table (a 65000 item table); or by examining the individual bit fields within the instruction word, which might take several steps.
- 5) Branch to the appropriate emulation subroutine to execute this instruction.

- 6) Execute whatever sequence of native instructions is required to achieve the same result as the emulated instruction.
- 7) Branch to (1) to fetch the next instruction.

These are the very same operations which every computer must use to execute a program. When implemented in hardware (such as an HP3000 CPU or a Spectrum CPU) several of these steps can occur simultaneously in different parts of the CPU. When emulated in software by a single processor CPU, each of these operations must occur sequentially one after another. A rule of thumb is that the interpreter overhead will cause an order of magnitude performance difference.

Consider the HP3000 NOP (No Operation) instruction. Typically, a computer can execute a NOP as fast or faster than any other instruction. In the emulator example above, the emulator will have to execute 6 native mode instructions to emulate a NOP. We'll assume that the emulated NOP (step 6) is skipped. This yields a best case expected performance of about 1/6th of an equivalent native mode program.

Of more concern is the expected performance of real instructions, not NOPs. Most of the instructions in an HP3000 program are LOAD and STOR. These instructions typically reference either the DB or Q registers. To emulate a rather complex LOAD Q+5,I,X instruction, some relatively sophisticated code is required. Remember that the +5 is embedded in the instruction word, not in a separate variable.

- 5.1) Extract the offset (+5) from the instruction into a scratch register.
- 5.2) Add the value of the Q register pointer to the offset in the scratch register.
- 5.3) Is this an indirect access (I)? If not then branch to (5.6).
- 5.4) Check for a bounds violation. This actually involves two steps. Is the referenced location greater than or equal to the TOS pointer register? Is the referenced location less than or equal to the DL pointer register? If so, then trap to a bounds violation routine.
- 5.5) Fetch the data from the referenced memory location into the scratch register.
- 5.6) Is this an indexed access (X)? If not, then branch to (5.8).
- 5.7) Add the value of the X register to the scratch register.
- 5.8) Check for a bounds violation.
- 5.9) Double the value of the scratch register because this is a word fetch, not a byte fetch.
- 5.10) Fetch the data pointed to by the scratch register into the scratch register.
- 5.11) Check that adding 2 to the top-of-stack (TOS) pointer will not cause an HP3000 stack overflow.

- 5.12) Add 2 to the TOS pointer register to make space for the LOADED data.
- 5.13) Store the data in the scratch register to the location pointed to by the TOS register.
- 5.14) Return to the main emulator loop.

This code sequence could be typical for emulating an HP3000 instruction. There are less sophisticated instructions. For example the stack operations such as ADD, SUB, DUP and DEL are not nearly as difficult to emulate as LOAD and STOR.

There are also more complex instructions which are executed frequently such as conditional branches, MOVE, SCAN and PCAL. PCAL (Procedure CALL) is particularly difficult. It has been stated several times that a program will be able to execute in compatibility mode and native mode and be able to switch back and forth. It has also been stated that several MPE intrinsics will not be rewritten in native mode until future releases of MPE-XL. When the HP3000 emulator encounters a PCAL instruction, not only will it have to route through the internal and external STTs, it must also determine if the called routine is in native mode or not. If it is, it must actually call it, otherwise it must continue emulating the HP3000 instructions.

MOVE and SCAN must actually execute as a loop in the emulator. These instructions will incur the usual loop overhead plus bounds violation checking during each loop iteration. The MOVE instruction incurs two bounds checks per loop iteration; once for the data source and once for the data destination.

Overall, using a software emulation approach to compatibility provides the capability to execute any HP3000 program on the Spectrum class machines. No re-compilation is necessary. The interpretation penalty is partially compensated for by the increased processor speed of the Spectrum machine.

The actual expected performance will depend on the ratio of compatibility mode code vs. native mode code. A typical HP3000 application is based on IMAGE data bases. Most of the time spent in such a program is spent executing IMAGE system intrinsics. There are not many instructions between calls to DBFIND, DBGET, and DBPUT. If the IMAGE intrinsics have been rewritten in native mode, then a program which spends most of its time in IMAGE code will execute nearly entirely in native mode. A good analogy would be a BASIC/3000 program which accesses an IMAGE data base. Using the BASIC compiler does not affect execution time very much because most of the time is spent in the IMAGE intrinsics.

Emulation or compatibility mode is only used until programs are converted into native mode. This is the case for MPE-XL code, intrinsics, HP system and applications programs, and for users' programs. Once everything is converted into native mode, compatibility mode will be obsolete.

The main argument against RISC computers has always been that they require many more instructions to accomplish a given task. Without improved compiler technology, this is true. Since each instruction gets less work done, more instructions are needed. Even though each instruction executes faster, there could be many more instructions to execute. More instructions also occupy more main memory. The success of a RISC based CPU depends on it's compilers. Incompetent compilers will doom any RISC based system to failure.

The compiler should be able to generate an efficient sequence of instructions, rather than just any old sequence. Often the compiler will make another pass (or passes) over the code in order to eliminate redundant instructions and other inefficiencies. This process is known as optimization. The current HP3000 compilers have no optimization pass, and in fact generate pretty sloppy code. The goal of an optimizing compiler is to generate efficient code. For now, "efficient" can be defined as compact, fast, and of course correct.

No one has seen HP's optimizing compilers yet, so there is no way to estimate how well they do the job. Most compiler optimizations have been well known and understood for quite a few years now. There's no reason to believe that the HP compilers will not do a good job optimizing program code.

But how can a programmer get the maximum possible performance from HP's new Spectrum computers? As always, programming in assembly language yields the best possible code, based on size and speed. The penalty for programming in assembly is severe. Programmer productivity is very low, and the chance of errors going un-detected is much higher than with a modern language such as PASCAL. Still, there are a few things to consider when programming in a high level language.

The HP3000 was based on a stack architecture with very few usable registers. HP's Spectrum computer systems are register based. Every time a variable is referenced, it must be brought into a register before it can be operated upon. If a program uses a large number of variables, they must be constantly moved back and forth from memory into CPU registers. Re-using variables for more than one section of code will allow the compiler optimizer to keep those variables in registers longer. Consider these PASCAL examples:

```
( swap some arrays )
for i := LOW to HIGH begin
    temp1 := a[i];
    a[i] := b[i];
    b[i] := temp1;
    temp2 := x[i];
    x[i] := y[i];
    y[i] := temp2;
end;
```

In this example, the compiler needs the variables I, LOW, HIGH, TEMP1, TEMP2, and pointers to the arrays A, B, X, and Y. These will require 9 registers. This is a trivial example, and in fact all of these variables could be loaded into the registers of the Spectrum machines. But it shouldn't be too hard to imagine a more realistic case where the compiler is forced to move variables in and out of registers too often. An optimizer can't easily tell which

variables are important and which are scratchable, so it must save them all. (In fact, depending on the effort expended by HP, it is possible for the compilers to look far enough ahead in a program to determine if a variable can be scratched, but it is a non-trivial procedure.)

Rewriting the same loop to use fewer different variables can reduce the register requirements of a program segment. Consider this rewrite of the previous loop:

```

{ swap some arrays }
for i := LOW to HIGH begin
    temp1 := a[i];
    a[i] := b[i];
    b[i] := temp1;
end;
for i := LOW to HIGH begin
    temp1 := x[i];
    x[i] := y[i];
    y[i] := temp1;
end;

```

Now only I, LOW, HIGH, TEMP1, pointers for A and B are needed at one time. This is a reduction from 9 registers to only 6. It is possible that in some situations, the duplicated loop overhead is less than the register load/unload overhead. If a block of code requires more registers than are available in the machine, then the compiler must swap the variables out to memory when not in use. Although the HP Spectrum machines have thirty-two registers, they are not all available for use within a program. Some are reserved for milli-code operations, some are reserved for the HP3000 emulator, some are reserved for procedure parameter passing, and some are reserved for CPU overhead such as stack pointers and program pointers.

Register allocation has never been a concern on HP3000 computers before because there were no registers to allocate. Every local variable is just as accessible as every other variable. In HP3000 promotional literature, this was touted as an advantage of the stack architecture. Each local variable could be accessed directly. But each variable access requires an indexed memory operation (ie Q+5). Register to register access is much faster.

Another feature incorporated into the Spectrum architecture is pipelined instruction execution. It is possible to begin an instruction before the previous instruction has finished executing. This may not seem to make any difference (and it doesn't when programming in a high level language), but it must be considered when programming in assembly language, or when writing language compilers.

Consider the actual operations of fetching an instruction from memory, decoding it, and executing it. These three steps require three sets of hardware.

- T+0) Generate the memory address of the next instruction, and read the contents of memory into a CPU register.
- T+1) Decode the instruction by routing the appropriate bits into gating and latching circuits.
- T+2) Those circuits which were activated now actually execute the instruction operation, perhaps requiring access to main memory.
- T+3) Main memory access for those instructions which require it. Otherwise begin a new instruction.

In non-pipelined machines such as the HP3000, each step would execute sequentially to process an instruction, and then repeat. In more recent CPU designs, including the HP Spectrum computers, pipelining attempts to reduce or eliminates each component's idle time. In this example, each CPU stage is idle for 2 out of 3 time cycles. By beginning the next instruction before the current instruction has completed, this idle time is eliminated, and throughput is increased. Consider the pipelined system in the figure below. The boxes represent the work being done by each processor stage during a given time period. Time is represented by T+n where n increases. The instructions are indicated by the memory location they are fetched from; M+n where n increases.

| | Non-Pipelined CPU | | | | Pipelined CPU | | | |
|------|-------------------|--------|---------|--|---------------|--------|---------|--|
| Time | Fetch | Decode | Execute | | Fetch | Decode | Execute | |
| T+0 | M+0 | | | | M+0 | | | |
| T+1 | | M+0 | | | M+1 | M+0 | | |
| T+2 | | | M+0 | | M+2 | M+1 | M+0 | |
| T+3 | M+1 | | | | M+3 | M+2 | M+1 | |
| T+4 | | M+1 | | | M+4 | M+3 | M+2 | |
| T+5 | | | M+1 | | M+5 | M+4 | M+3 | |
| T+6 | M+2 | | | | M+6 | M+5 | M+4 | |
| T+7 | | M+2 | | | M+7 | M+6 | M+5 | |
| T+8 | | | M+2 | | M+8 | M+7 | M+6 | |
| T+9 | M+3 | | | | M+9 | M+8 | M+7 | |

Of course pipelining is not quite as simple as this graph would imply, but it does give the right idea. In the non-pipelined system, the entire CPU must wait for the completion of each instruction. In the pipelined system, more of the CPU is kept busy more of the time. The pipelined processor in this graph is not faster than the non-pipelined processor, but at the end of a given period of time, it has executed more instructions. If each time period in the graph were one microsecond, how would each processor be rated for speed? The non-pipelined processor executes 1/3 MIPS (Million Instructions Per Second). Rating the pipelined processor is much more difficult. There is no correct rating. Any of 1/3 MIPS, 1.0 MIPS, or "up to" 1.0 MIPS might be used. It will depend on the method chosen by the manufacturer.

The real world is also a bit more complicated than pictured here. Also, this example of pipelining shows no provision for interrupts. Where does the processor restart after the interrupt? After the instruction which has most recently been fetched? Or after the instruction which has finished execution? What if some stages perform partial instruction execution? Some pipeline implementations have 5 or 6 stages, not just the three shown here. The additional stages are used for indexed memory accesses and for faster memory management. In this example there is no mechanism for accessing main memory during the instruction execution phase. If the execution stage requires access to main memory, then the instruction fetch will be forced to wait.

This is another advantage of the register based CPU. As the ratio of register to register instructions increases, so does the pipeline efficiency. The more often the processor must access memory, the more often the pipeline will be kept waiting.

If pipelining is such a boost to throughput, and pipelining has been understood for years, why hasn't Hewlett-Packard incorporated pipelining into the HP3000 models? The problem is memory access. Nearly every HP3000 instruction accesses main memory several times during its execution. Consider the same `LOAD Q+5,I,X` instruction as before. The CPU must (1) fetch the instruction; (2) fetch the pointer at `Q+5`; (3) fetch the data at `Q+5,I,X`; and finally, (4) store the data at the top of the stack. There is very little spare time to pre-fetch the next instruction. In fact the dual ALU (Arithmetic Logic Unit) of the HP3000 series 64 was added to perform some of the steps listed earlier simultaneously with each other. For example, a bounds violation could be tested for in one cycle instead of two. A single ALU must test a memory address against the TOS register, then against the DL register, a two step process. With two ALUs, the series 64 executes both tests simultaneously. This eliminated one possible free time slot to fetch another instruction, but increased the series 64 speed significantly.

The pipelining mechanism is the cause of some peculiarities in the operation of the Spectrum processor. One publicized "property" of the Spectrum processors is the so-called "delayed branch". Not only is this a mis-nomer, but nowhere in the literature, where delayed branches are mentioned, is the cause for them given. We are left to wonder why they were ever mentioned in the first place.

Delayed branches are the result of a pipelined CPU. By the time that a branch instruction is decoded, the instruction following the branch has already been read into the CPU. Rather than waste a perfectly good memory cycle, the CPU designers decided to finish out the cycle

and execute the instruction anyway. Thus, the instruction after the branch instruction is always executed. More traditional pipelined processors simply flush the pipeline whenever a transfer-of-control instruction is encountered. Consider the graph above. Suppose that the instruction at M+4 is a branch instruction. By the time the CPU decodes the branch, the instruction at M+5 has been loaded into the pipeline. The execution of the branch can prevent the fetching of M+6, but M+5 will still be executed.

This phenomenon is taken into account by the compilers transparently to the user. Only assembly language programmers and compiler writers need consider the delayed branch when coding a program. A NOP or similar instruction which will always be executed, must be placed after the branch instruction. In a recent HP Journal article, the case was described where HP's optimizers can replace the NOP with a useful instruction. One restriction on the instruction after the branch is imposed by the processor. It must be a single cycle instruction, and cannot be a transfer-of-control instruction because the processor will already be fetching the instruction at the branch target address.

The same phenomenon will occur when calling a subroutine. By the time the CPU realizes that a call instruction is being executed, the instruction after the call will have been fetched into the pipeline. For whatever reason, HP has chosen to glamorize (or at least publicize) delayed branch instructions, but has not mentioned if calls are "delayed" or not. If the Spectrum instruction set is regular and predictable, then calls are probably delayed as well.

All in all, time will tell how successful the new HP3000 high end computers will be. Many of the new architectural features of the Spectrum processors are clearly included so that HP need never again say it's sorry (particularly the 32 bit plus 32 bit address capability). The HP3000 64 Kbyte data address limit has not been winning many latter-day followers in the light of competitive minicomputers.

Hewlett-Packard has not leap-frogged anybody with their new computer systems. But with new high performance, pipelined processors, large memory capacity, optimizing compilers, relational databases, true virtual memory (VM), symbolic debuggers, and HP3000 family compatibility, HP is finally keeping up with the Jones's.

There are two very good books available which describe the trials and tribulations of designing a new computer system. They can provide a view of the world from the other side of the fence. New computer systems don't grow on trees. Designing, building and programming them is a monumental task. One of the books is The Mythical Man Month (Fred Brooks). Fred Brooks was the project manager of the IBM OS/360 project. The book describes some of the things which happened during that project, and the lessons learned. The Soul of a new Machine (Tracey Kidder) chronicles the group of engineers who designed the Data General Eclipse computer systems. Both books are very good reading for a project manager, or for someone who has had to put up with project managers. Both are commonly available at public libraries and bookstores.

Jeff Hecker is a system programmer for Arens Applied Electromagnetics, working on the "Presentation Graphics" package, and has been involved with the HP3000 and the MPE operating system for the last six years. He also has been responsible for moving, debugging and optimizing several applications from the HP3000 to other computers and operating systems, and back again. Prior to working with the HP3000, Jeff was a digital hardware design engineer for a Maryland based telecommunications company.

4GL - The Controversy Rages On

Karen Heater
Infocentre Ltd.
6303 Airport Road
Suite 300
Mississauga, Ontario
L4V 1R8

Introduction

I truly believe that the question which we are asking ourselves regarding 4th Generation languages is quickly changing from "Should I?" to "Which one and how do I make it successful?".

After speaking at a number of Interex Conferences and local area user groups on various topics surrounding 4GLs, the feedback from you, the Data Processing Professional of today and the future, seems to bear out this belief.

The goal of this paper is to assist you in selecting and successfully implementing 4th Generation Development Software. The paper will touch on the strengths of 4GLs, their potential dangers, a few suggestions on how best to approach your product evaluations and lastly some things that must change if you are to be successful.

I hope that at its conclusion you will feel more equipped to make the best decision possible for both you and your organization.

4GL - The Controversy Rages On

A Rising Need

The emergence of 4th Generation Languages and their associated alternative development methodologies grew out of a clear and definite need arising in the data processing industry.

Let me take a moment to explain -

A recent computer industry forecast predicted that the number of installed computers would increase by a factor of ten in the next ten years.

A Computerworld survey indicated that the number of computerized applications in existing data processing departments is growing at a rate of 45% per year.

In Computerworld it was also estimated that \$10.00 was the average cost of one debugged computer instruction.

Last year, North America produced more computers than programmers and science graduates combined.

And finally, it is also generally accepted that the average backlog of applications in North American DP departments is between three and four years. Visible backlog, that is.

It becomes blatantly apparent that some drastic changes are necessary in our approach to the development of application software if we are to be able to deal successfully with existing and future growth.

The problem we are faced with then is how are we, to be successful, not only to survive the next decade, but to conquer it, without the necessity of creating programmers out of every person in our organization.

Enter 4GLs

For a start it seems logical to address the productivity of the DP Professional. Helping the DP Professional to develop systems more quickly would definitely have significant impact on current growth, the backlog and the influx of future needs.

What is needed then are tools to assist the DP Professional in doing his or her job. Tools that would take care of most of the mundane and repetitive aspects of developing production application systems. Tools that "understand" what is involved in the fundamental functional areas of an application; i.e. data storage structures, menu driven end-user

4GL - The Controversy Rages On

interface, online data collection screens, online and batch reporting, transaction processing, application security, etc. and is able to assist the DP Professional in pulling it all together.

Traditional 3rd Generation Languages are unquestionably logic driven. The constructs of the language are put together to form logic to perform screen handling, reporting, etc. These languages do not easily lend themselves to a transformation whereby they become development assistants. Virtually all of the "smarts" of application development must reside with the individual because the language itself does not truly provide for this.

4th Generation Languages have been loosely defined by various individuals on various occasions as being "products which produce results in one-tenth the time of Cobol". In order to ensure this productivity increase, these languages are designed to be more action than logic driven. In other words, when developing an application, one uses the constructs of the language to state what is needed i.e. screen layouts, report layouts etc., not how logically to perform the task. The language itself handles the fundamentals of interaction with the data storage structures and dictates a certain standard interface for the user.

These languages by nature, assist the DP Professional by providing intelligent defaults throughout the development cycle and imposing certain application design standards. (We will discuss the benefits of these standards a little later.)

What have emerged then are a new breed of development language which when used properly (this cannot be emphasized too much) will significantly affect the individual DP Professional's productivity.

But Wait, There is more

The crux of the problem which appears to be facing us for the decade to come seems to lie in the "development bottleneck", our inability to implement solutions to satisfy all the user needs in a timely manner. The failure to provide these solutions begins the steady build up of that infamous 'backlog'.

Backlog

I'd like to digress for a moment to discuss this 'backlog' that is so often found in discussions of the need to improve productivity.

Backlog results from a development bottleneck. Backlog is catastrophic both from a morale standpoint and its effect on the progress of computerization within an organization.

4GL - The Controversy Rages On

We have often heard of backlog described as being comprised of two parts - the visible and the invisible. Visible backlog is readily identifiable, we can describe and count the requests for enhancements or new development that we are just not capable of getting to right now. Visible backlog is difficult enough. We all know that an in-tray that continues to grow despite our dedicated efforts to employ it, is not the most uplifting of experiences. It would seem enough to resign ourselves to a certain quantity of visible of backlog, as we do a certain level of paper in our in-tray. Unfortunately there is still the probelm of Invisible backlog.

Invisible backlog is the wealth of things that users stop asking for. And further down the road, it is the things that users stop thinking about.

The fact that users stop asking may appear a blessing. We may even attempt to rationalize that if they stop thinking about it, it relieves them of their own frustrations. But users are the people who run the daily operations of our organizations. They build products, sell them, collect payment for them and as a result the business makes profit or some variation on this theme. It is they who can tell us how to help them to perform their function more efficiently so that business can grow. The loss of their input, their ideas, over the long term is devastating.

Without a doubt the 4th Generation Languages that exist today are capable of dramatically improving productivity. You do not need to take my word for this. Above and beyond an intellectual discussion of their merits and potential, we as vendors are always more than happy to provide you with an extensive customer reference list of HP3000 and 4GL users who will discuss their experiences with you.

Certainly, improving each DP Professional's productivity by providing products such as 4GLs is a valid start and possibly even the most far reaching solution. There are two other areas though that should be reviewed in order to completely address the issue of maximizing productivity in an attempt to successfully meet our needs now and in the future; Packaged Software and End-user Computing.

Packaged Software - Ah, the plight of packaged software. At one time Turnkey Systems abounded. A basic package, a little customization, a friendly unassuming minicomputer and voila, a smooth entry into the world of automation. As educational institutions added "Computer Science" to their curriculum, a wealth of programmer/analysts converged on the marketplace and there appeared a growing "computer literacy" across the nation.

4GL - The Controversy Rages On

What resulted was a feeling that it would be more cost effective to employ one's own programmers to develop software that truly would meet the unique needs of the organization. The movement to bring development and maintenance in-house was on.

The growing popularity of productivity software, namely the 4GL has in the past caused many people to ponder the question, "if in-house DP Professionals can produce applications at least 10 times faster now then why bother to purchase packaged software at all?".

Well the answer to that is quite simple, but it is still surprising the number of organizations that purchase a 4GL and unnecessarily and most importantly, unwisely, begin to 're-invent the wheel' over and over again.

More realistically, 4GLs and Packaged Software make a beautiful team. Your packages often become the building blocks of your system while the 4GL allows for the customization, enhancement and integration.

End-user Computing - I can imagine that the same shiver of apprehension that just went up my spine may very well have gone up your spine too.

The enormous growth in popularity and sophistication of the microcomputer workstation has made this an area worth discussion.

Many requests from users that traditionally required one-off customized reports can now be adequately handled by the user, through a combination of electronic spread sheet and graphics software available for the PC. The major area of concern seems to be the accessibility of corporate data to the users of this PC based software. This is a whole area of discussion unto itself and it is more sensible here to direct you to the many good papers available that discuss this topic in-depth. In fact, I have one myself should you be interested.

To recap then, an approach to providing solutions that combines -
 Productivity Software (4GLs)
 Packaged Software
 End-user Computing
 will put you well on your way into the next decade successfully.

Watching Out

I hope at this point you feel comfortable with the role of the 4GL. It certainly appears that their use will be the most effective solution in our approach to data processing in the next decade.



4GL - The Controversy Rages On

The 4GL is new and as such there is still much to learn about how best to take advantage of its power. From the experiences of those who have already made the plunge, come the following potential dangers that it would be wise to be aware of so that your implementation can be as smooth and successful as possible.

The real danger of a 4GL is in trying to turn it into your entire solution, i.e. using the 4GL to re-invent the wheel rather than investigating the availability of packaged software.

A 4GL is not necessarily, and most often not an end-user tool. Suddenly providing an end-user with access to your 300,000 customer records so that he or she can produce for themselves that "very simple" report can become a disaster. Do they know what an 'Image Key' is and how one can just provide a value for it rather than serially reading through each customer record looking for the one they want? One request from the backlog may be removed but someone has to handle the potentially serious problem of system performance degradation.

Another danger of the 4GL lies in the DP Professionals approach to its use. A 4GL is dramatically different by nature than a 3GL. The basic approach or methodology is different. 'Translating' an application from a 3GL to a 4GL will not yield the most efficient implementation of the application. It is important to re-think the application, as if you were developing it for the first time, using the 4GL.

4GLs inherently and by design will cause an increase in the speed of development of production systems. Two things to watch out for here are; impact on data and the implementation bottleneck. A little later under 'managing the 4GL' we will discuss how to minimize both of these.

Before we move on to a brighter subject, that of the benefits associated with the use of a 4GL, there is one last area that should be reviewed, that of system performance.

It is not necessary to accept significant performance degradation in return for your productivity gains. There are a number of extremely good 4GLs on the market today and one in particular, that is known for its "Speed" on the system. I'd like to take a moment to discuss a common misconception that seems to be prevalent with respect to performance before we move on.

4GL - The Controversy Rages On

Performance

I have often heard it asked "If I buy this 4GL will I need to buy more memory and disc in order to use it?". There are really two answers to this question, 'not necessarily' and 'possibly'. Confused? - let me explain.

A good 4GL does not create the need immediately for more power or resources. But, a good 4GL will probably create the need for more resources within 12 months of its purchase.

4th Generation applications can and do perform well in production environments. It is not typically the addition of a 4th Generation production application that chews up system resources, it is the constant use by a staff of DP Professionals over a reasonable period, let's say one year, that does.

When you consider the quantity of maintenance and new development that will have occurred it is obvious that the HP3000 is within one year, handling what would previously have been 3-5 years of work. It is this volume of new systems and improvements to existing ones that has caused the need for more power and resources.

The awareness of the impact of the dramatic increase in the volume of applications on your current system is the most crucial issue here.

The Benefits

A 4GL is a powerful tool that must be managed properly in order that it be an effective one for your organization. When managed properly there are significant benefits to be accrued by the DP Professional, the user and the organization as a whole. It is because of these benefits that each and every HP3000 DP department should be taking a serious look at the acquisition of a 4GL.

* Savings, Savings, Savings. Increasing the productivity of existing DP Professionals could potentially alleviate the need for additional personnel. Applications both backlogged and new requests, can be put into production at least 10 times more quickly giving users access to systems to better assist them in their daily activities. The more productive the users, the more productive the business.

* Controlling Maintenance. The nature of a 4GL being predominantly action rather than logic driven, means that the code itself is much less and more concise. It is easier and less costly to maintain 1000 lines of code than 10000. As applications are put into production faster, there is a better likelihood that they will meet the immediate needs of the user. As a result, less time is needed to perform

4GL - The Controversy Rages On

maintenance tasks that are just bringing a production system up-to-date with the current needs of the user. The cause of these dramatic reductions in maintenance is the use of the prototyping or protocycling approach to systems development when using a 4GL.

* **Morale.** It is without a doubt most satisfying to work in an environment where you are perceived to be doing your job well. By putting solutions into place quickly for users, the DP Professional feels accomplishment and users are much more confident about the DP Department's ability to respond and satisfy their needs.

* **Standards.** As we discussed earlier, the 4GL assists the DP Professional by imposing a variety of design standards. Two applications developed by different people, with a good 4GL, will look very similar if not almost identical to the user. The system will function in the same manner i.e. menus, screens, prompting, message display. The time needed for a user to become productive on a new system is minimized. The DP Professional can then be spending the time previously spent on training the user, on new development. These standards manifest themselves also in the code itself. Not only do the two applications function the same but the code will be very similar. Readability and standardization of code account for enormous savings in development and maintenance time.

It is obvious that despite the potential disaster areas in using a 4GL there are tremendous benefits in using the right one, properly. I'd like to devote the rest of this paper to the two issues, Choosing the right one and Managing its use effectively.

Choosing the right one

There are really three different levels on which your short list of 4GLS should be evaluated -

1. Functionality / Performance
2. Comfort
3. User acceptance.

Functionality / Performance - It is at this first level that the different 4GLs need to be evaluated to determine if they are capable of being used to put together the type of production applications you need, that will execute with what you consider acceptable performance.

Acquiring and using a demonstration tape and visiting or speaking with reference sites are probably the two most fundamental tasks which should be undertaken at this stage. Both are time consuming but there is really no alternative.

4GL - The Controversy Rages On

Comfort - Each 4GL has its own personality. When designed the individuals responsible had their own ideas about what the 4GL should do well and how the average DP Professional would use it.

There will probably be one style or personality that you and your personnel will feel the most comfortable with. This is very important because ones comfort and confidence in using the 4GL will determine the degree of productivity improvement that will be attained.

It is through the use of the trial tape that you can assess the effectiveness of the vendor's support services, another very important aspect of the 'comfort' in the use of the 4GL as there is a large learning curve which everyone will have to go through.

User Acceptance - If the 4GL does everything you want it to, the DP Professionals love it but the users can't stand how the applications look, we have not necessarily accomplished anything.

Remembering that often the bulk of defaults or standards that the 4GL imposes will be in the end-user interface area, this could indeed become a problem. During the trial tape period is a good time to put together something which a number of key users can get their hands on and evaluate. A trip to a reference site along with the DP Personnel would probably also be beneficial.

Some Tips

Just a few things to look for that can often be overlooked when evaluating, a trial tape but are quite fundamental to a successful implementation -

How well does the 4GL integrate with your existing environment, i.e. programs, files, security, etc.

How easily does the 4GL allow you to access subroutines in 3GLs. (Don't forget that Fortran pricing routine developed by someone in 1968 that you must use).

What tools / aids does the 4GL provide to assist with implementation. i.e. end-user documentation (Don't forget the development bottleneck will quickly be replaced with the implementaion bottleneck)

Effectively Managing the 4GL

Choosing what appears to be the best 4GL for your organization is an important and crucial first step. Now all you have to do is get it implemented and then the new systems that will be developed, implemented.

4GL - The Controversy Rages On

Having made the purchasing decision I'll assume that your selling job to management, users and DP Personnel has been successful. You now have a product and alot of people have alot of expectations and probably some unspoken reservations.

Management

Let's start with management. Technological advances that allow machines to perform significantly better are commonplace and easily accepted. Management tends to be far more sceptical about claims of dramatic improvements in people productivity. Their stop watches will be out and running soon after the purchase order leaves the building.

An important aspect of effective management of a 4GL is the setting of expectations. Management should be clear on the fact that it will take a number of development cycles before the true productivity gains can be realised. Have them put away the stop watches for at least the first 6 months, giving the DP organization time to settle in.

Be conservative. It is much more fun to exceed expectations and be hero later.

The Users

Earlier in the paper 'Prototyping' was mentioned. Briefly, the nature of the 4GL lends itself well to a development methodology where the users and DP Professionals interact on a regular and on-going basis to work an application from an initial prototype to a fully functional production system. This methodology requires the commitment by knowledgeable users to the development process.

Again we are speaking of expectations. Constant dialogue between DP and users was one of the first things to go as a result of the lengthy development cycle typical with 3 GLs. This breakdown in communication is the real culprit behind the devastating problem of invisible backlog.

It will take time and practice to put this dialogue back into place, to re-open the lines of communication. The commitment must be there. If it is not, applications will be developed more quickly, but they will still not meet the real needs of the user.

4GL - The Controversy Rages On

The DP Department and Professionals

Finally, the DP Department and the DP Professional. Being asked to play with a couple of 4GLs as a short term project is one thing but being given a 4GL and being told to "chuck" Cobol is quite another.

"Keeping an open mind" as an accepted philosophy will be your only chance of survival.

During the evaluation cycle the following areas that pertain to the DP department and/or each DP Professional should be addressed;

1. Data security and control
2. Development Methodology
3. Fear.

Data Security and Control - As the development cycle speeds up and your system becomes populated with new applications and all of their associated files, it is extremely important to have databases and files organized and documented. If they are not, you could find yourself in a reactionary state dealing with one disaster after another. The 4GL will have effectively take control of you.

Development Methodology - 4GLs are not best used with the development methodologies that we have spent years mastering using 3GLs. Frozen specifications, long development cycles, masses of program documentation and maintenance of analysis and design bugs are a thing of the past. Prototyping / Protocycling is the development methodology that is required with a 4GL. I would recommend that you take the time to review all current literature on the subject of prototyping.

The point to be made here is that your current methodologies will not work with your new 4GL. A full understanding of and adherence to a new methodology must occur early on with the 4GL.

Fear - In reality the average DP Professional has spent his or her formal education and years of practical experience refining the skills of application design and development using a 3rd Generation Language. They are competent and confident in their abilities. The implementation of a 4GL means that there is a new learning curve to be addressed and many of the skills, so finely refined are not applicable in this new approach.

4GL - The Controversy Rages On

There is potentially fear for ones job and alleviating that, fear of success in ones jobs with the new tool.

Effective use of a 4GL does require an emphasis on some skills that were not as important before - namely communication skills, as the bottleneck shifts from development to implementation.

It is important for DP management to take a good look at the people in their department and individually evaluate the level of re-training each person will require and prepare the appropriate education.

Starting off with the straight forward applications will go a long way in assisting everyone in managing the new learning curve, building confidence, nailing doing development guidelines and bringing users back into the development cycle.

In Summary

4GLs are serious and valuable new tools to assist in managing the requirements of the Data Processing Department in the decade to come.

Much thought and care must be taken in their selection and implementation - to ensure their success with your organization.

Most important is the setting of realistic expectations and the understanding that a 4GL will inherently change the nature of how DP professionals do their job.

If managed effectively the acquisition of a 4GL can mean tremendous savings and a much needed breath of fresh air.

INFORMATION CENTERS AROUND 4GLs

ROBERT REMILLARD, BOB STANLEY
COGNOS INCORPORATED
3755 RIVERSIDE DRIVE
OTTAWA, ONTARIO, CANADA K1G 3N3

Information is probably the single most valuable commodity in the world. Without sophisticated electronic information processing systems the world's financial systems would collapse, business and industry would quickly grind to a halt, and governments would be unable to function. We live in an information-dependent world, and the dependency is growing rapidly.

The "information explosion" that has characterized the last quarter-century shows no sign of slowing down. Until recently it has been fuelled largely by the hardware end of the information technology producers. Each new generation of computers is smaller, more powerful, and less expensive than its predecessors. More importantly, this onrush of technological development has made extremely sophisticated technology widely accessible to people who have no specialized training in the use of computers or in the basic concepts underlying data processing and retrieval.

This paper will look at the causes and effects of these changes in the world of MIS and their implications. It will examine the use of software tools, and particularly fourth-generation languages, in attempting to find solutions to the new challenges to MIS. These challenges include the emergence of information center tools and the simultaneous growth in the numbers of non-technical users. Our objective is to present ways to integrate the needs of end-users, the MIS specialists, and the corporation as a whole through the use of fourth-generation languages and information center tools.

In the Beginning...

Twenty-five years ago there was the mainframe. Large, complex, and very expensive, it was the exclusive domain of an elite cadre of highly specialized engineers and technicians. Even the machine's physical environment had to be meticulously controlled. The vast majority of people in the workforce had no access to the computer, and were generally unaware of its impact on their lives, or even of its existence. All processing was done in batch, so there was no need for interaction with users. For many of the veterans of the data processing industry, those were the good old days ... when good old computers scared good old boys!

Then came the minis; the HP3000 first saw the light of day. The advent of the minicomputer had several significant results. It made possible the purchase of computing power by much smaller corporations and organizations, by department stores and public libraries, hospitals and universities. Within larger organizations even individual departments could sometimes afford their own computer. The minicomputers also introduced interactive processing. Thus the computer became more available, but at the same time the fragmentation had begun.

As the mini made the use of computers for information processing more widespread, it also made the computer more visible - there were screens popping up on people's desks. People became aware of the impact of computers as they increasingly came into contact with the effects, both good and bad, of computerization. Before long it seemed that everything was "on the computer", which meant that to get the information they wanted, people had to go to the MIS department. Often they didn't just want information, they wanted whole systems to handle information input and output.

This new awareness created tremendous pressures on the people who worked in MIS. Programmers and analysts were very much in demand. But increases in staff still couldn't meet the demands placed on MIS. So the backlog was born, and the only thing growing faster than the backlog was the frustration level. The good old days were over, and suddenly the word "computer" became a synonym for everything that could go wrong in the organization. Remember the old joke: To err is human, it takes a computer to really screw things up.

Then along came the newest revolution: the personal computers. These started as hobbyist machines, and made it to the business world when spreadsheets and analysis tools appeared. Now the manager, the clerk, the accountant, the vice-president, the secretary, and the personnel manager can all have their own computers and by-pass MIS. Most of them are stand-alone machines, or more recently small clusters using a variety of local area networks. They are not necessarily compatible, either with each other or with the minis and mainframes in MIS. But at least all the end users have their own information processors.

The trouble is they still don't have the information. Or, more precisely, they don't have direct access to the information residing on the corporate machines. They have word processors and spreadsheets, database management systems, report writers, and graphics generators. But the corporate data, that precious mother lode of information that they all want to mine with their electronic tools, is still up there

on the "big" computer. The crux of the problem is that information has no intrinsic value. Its value is directly related to its availability, and that implies having direct access to the right information at the right time.

The World As It Is...

The purpose of this ramble through the recent history of the development of the information processing industry is to help provide a perspective for what has been happening in the more immediate past. In the 1980s the emphasis has switched from hardware to software. There are fourth-generation languages, there are information center products, expert systems, and more, all claiming to offer the solution to the information gap problem.

The overworked people in the MIS department are told that the tools they've used for so many years are obsolete. Whether it's COBOL or FORTRAN or RPG, it's too slow and out-of-date, and the end users can't understand it. Of course these third-generation tools weren't meant to be understood by end users. And they can't simply be thrown away, because most of the organization's applications have been built with third-generation software technology. So if software is the solution, how is it to happen?

For the beginnings of an answer, it is necessary to take a closer look at the real needs of all those involved in the system. Essentially there are three groups involved: the users, the MIS department, and the corporation or organization itself (see Figure 1).

The users, as we have seen, need effective information management. That means fast, easy access to the kinds of information they must have in order to carry out their work. Their problem is that too often they can't get exactly the information they want. Or they get the right information, but it comes weeks too late to be of any assistance in decision-making. Micro-based tools don't give the users access to the central database; they don't allow the user to transfer applications or data to and from corporate files; and they cut the user off from the expertise resident in the MIS department. In short, they isolate the user from the mainstream of information in the work environment.

At the other end of the needs scale sits the corporation. The corporate requirement is twofold. First there is the need to maintain productivity at an optimum level. This means giving people access to the information they need in the most efficient manner possible. On the other hand corporate data is an extremely valuable resource that must be protected from unauthorized access. It is also important to ensure that legitimate users of the system are not corrupting the

database with inconsistent data, nor bringing the system to its knees by overloading it with unnecessary requests.

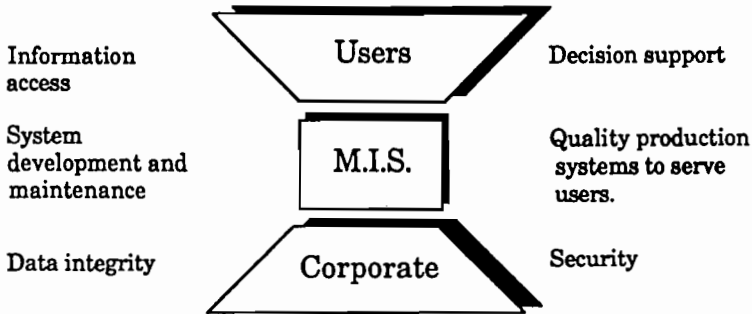
Squarely in the middle between these two sets of needs is the MIS department. The first function of MIS is to service the users' needs for timely access to information. At the same time, they must play the role of guardians and maintainers of the corporate information resource. Their job is to preserve data integrity and maintain the systems. At the same time they must try to meet an explosion of demands from the users that result in an ever-increasing backlog. Whether they attempt to meet the demand by developing their own applications with third-generation tools, or by buying packaged applications, chances are that what they provide will still fall far short of what the users want.

This is usually because the users and the MIS staff tend to work in isolation from each other; there is rarely any interaction between the two groups during development. So the only goal that is achieved is that the database remains secure. Largely inaccessible, but secure.

This situation comes about because the software environment is built around mostly third-generation languages. Let's take a look at this environment. At the functional heart of the computer lies the operating system. MPE is the operating system for business environments on the HP3000. All users must understand how to interface with MPE to some degree, even if only how to log on and log off.

Separate and distinct from the operating system is the file management system (or systems) which is used to read from or write to application data files. For an HP3000 running under MPE, Hewlett-Packard provides IMAGE, KSAM, and MPE file management systems. To create an application, a set of programs is written which must be able to communicate with the appropriate file management systems. In addition these programs may have to "talk" to the operating system in order to perform certain operations.

The problem with 3GLs is that they have very little, if any, knowledge of either the file management systems or the operating system. As a result, the programmer must intimately understand all these systems, as well as the programming language, in order to create a production application. The process is enormously time-consuming, and obviously does not invite the participation of the eventual users of the application in the development process.



Fourth-Generation Languages

What is needed to resolve the impasse is a set of tools that can reduce the MIS workload while simultaneously meeting the requirements of the users for access to timely information that is compatible with their micro-based systems. It should also be possible for MIS to quickly develop applications in response to new or changing requirements. And of course the corporate need for productivity and security must be met.

The widely touted solution is the fourth-generation language - or the 4GL. There are at least a dozen serious 4GLs available for minis. They are basically of two types: there are information center 4GLs and there are development center 4GLs.

A true development center fourth-generation language such as PowerHouse, the most widely installed 4GL on the HP3000, understands and communicates directly with the operating system, minimizing the knowledge required to use it. It should provide an intelligent interface to the file management systems, enabling it to read from or write to any of the files it can access. Yet it must be independent of the file systems, so that it can easily take advantage of new file structures. A 4GL must also be able to access and manipulate data that are already maintained by standard utilities or third-generation languages so that existing applications are not rendered immediately obsolete. The real test of the development center 4GL, however, is in the richness of the syntax. It must allow the development of real-life, complex business applications, and have little or no restriction on formatting and design specifications. The overall effect of such a language is to open up the closed MIS world. Applications can be developed and prototyped in a fraction of the time. Now it is possible to involve end users in the process, and to modify the

applications to meet their requirements following periods of testing and development. Maintenance time is similarly reduced, and for the first time the users can speak the same language as the MIS people. With improved communication comes better products and increased productivity, as PowerHouse sites will confirm.

Many of the products being marketed today as 4GLs do not meet these requirements. Often they consist of little more than a proprietary database management system surrounded with simple retrieval facilities. These are what are generally known as "information center" 4GLs. They are designed to simplify data retrieval for inexperienced or untrained users, and they generally do this very well. Typically they provide a user friendly "hand-holding" interface that guides the users through the development of a simple screen or a report. Within certain limitations, they can be quite effective. However, there is a price that must be paid for this simplicity. It is this: information center 4GLs inevitably sacrifice functionality and efficiency in favour of ease-of-use.

The trade-off is power for friendliness, which may be OK until you try to use the information center 4GL to build a large production application. Then the limitations become more evident, and the programmer's job more challenging, to say the least. The fact is that this type of information center 4GL is not designed for developing full-scale production applications. So it is not effective beyond meeting the immediate information needs of a small group of users. It does not address the broader problem of the isolation of the MIS group from the users they serve, and it may well increase the fragmentation because it neither replaces nor communicates with the third-generation environment.

The Development Center Concept

How, then, to take advantage of the obvious benefits of the information center concept while addressing the need for greater integration? We propose a total system in which a true "development center" 4GL acts as a front end to the operating system and file management systems, and provides the MIS department with vastly increased development power, while at the same time it links to a series of modular information center products that in turn are linked to each other. Such a system requires further explanation.

Central to the development center 4GL is an intelligent data dictionary. This is a central storage place for file and element definitions and their relationships. It is the key to centralization and standardization, but it should also be a great deal more. The data dictionary should understand the

INFORMATION CENTERS AROUND 4GLS

structures and relationships it defines, and be capable of working with and enhancing the functionality of different file systems. It should assist programmers in setting up security and creating applications, and give them a logical view of the system's files. Applications created in a system driven by such a dictionary use far less code, and are much easier to maintain.

Where the information center 4GL forces the user to follow a predefined series of commands, the development center 4GL uses high-level non-procedural statements designed for use by all levels of users. The language interprets these statements and makes assumptions about the operation being specified. This makes it very easy for non-DP users to use, especially in areas such as report writing. But the professional programmer needs more flexibility, so it is possible to override any of these assumptions with more detailed options or procedural constructs and verbs. The ability to override the choices made by the language is one of the hallmarks of the true development center 4GL. It creates a multi-level environment in which all kinds of users can function effectively.

Must Have Power

The development center 4GL recognizes that the vast majority of serious application development is carried out by the DP professionals in the MIS department. The language is therefore designed for these professionals, and it has what the information center 4GL lacks - **power**. The true development center 4GL must have the power to enable you to build as complex an application as you can think of. Yet the language is also designed in such a way that it can be used by non-professionals. What this means, in fact, is that the users can become actively involved in the development process in one of two ways:

They can develop their own limited applications using the system defaults. Because they are now working in the same language, such applications may be further developed by MIS if necessary and emerge as full-scale production applications. In such an environment, a major role for MIS is to help in the data modelling process early in the prototype to ensure long-term viability of the system. This can only be done around a powerful data dictionary.

Alternatively, they can work closely with MIS during the development and testing of prototype applications. Because the 4GL reduces the coding required by as much as 90 percent over 3GL products, it is feasible for MIS to produce "quick and dirty" prototypes which can be tested by the users and refined before a final production version is produced. This helps to overcome the problem facing every system designer -

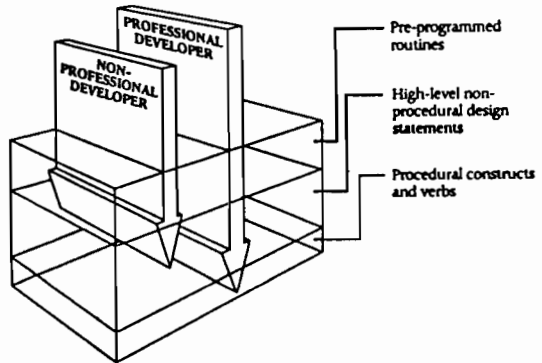
finding out what the users **really** want. The fact is that the users usually don't know what they want until they can try something and discover what they don't want.

Both these techniques not only save MIS a good deal of time in terms of application development and maintenance, they also go a long way to meeting the users' needs, thus meeting one of our requirements, that of increased productivity, in two essential areas. Perhaps most significantly, they help to bridge the gap between MIS and the users.

Powerful defaults

High level statements

Meet user levels of expertise



The Information Center Concept

However, there are a great many non-technical users who either can't, won't, or don't have the time to develop their own applications. The sales manager who wants a quick run-down of the latest sales figures from each of his regions. The vice-president who wants to know what capital equipment is held by each of the departments under her control. The financial planner who needs facts and figures to put together a budget presentation. There are hundreds of such requirements in any corporation. These people basically need to extract, manipulate and format data. Because they have little or no computer expertise, they need information center technology.

A software tool is considered to be in the information center category when it is easy enough to use that it can readily be used unassisted and with minimal training by end-users with no technical background. Whether these decision-support tools run on minis or on micros is not important. It is important that they can directly access and use the corporate data, eliminating the need for repeated re-keying.

In other words, the information center must be integrated with the development center. If end-users can use simple tools to obtain the data they want, this eliminates a great deal of the information retrieval burden currently carried by

the MIS department. For example, a user-friendly reporting tool can meet most of the end-users' needs for on-line queries and printed reports. A financial planning tool can provide the number-crunching ability managers require for budgeting and financial analysis. These and other information center modules must be able to transfer data to and from the database controlled by MIS in the development center. They must also be able to communicate with each other, and transfer data back and forth so that a financial report can include text from the report writer. This is the real challenge of information center tools: how to integrate them with the development center.

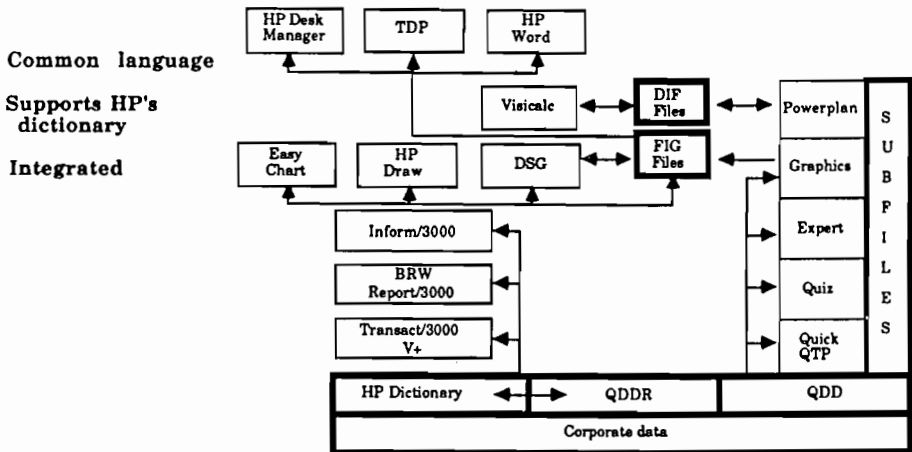
This is where the power and flexibility of the development center 4GL really show their value. We have already referred to the importance of the data dictionary as a central repository of data definitions and entity relationships eliminating concerns about data integrity. The dictionary also plays a key role in integrating the different levels of needs. Using a state-of-the-art dictionary such as PowerHouse Dictionary, the MIS department can establish security for different groups of users through the dictionary, not just at the application level but also at the level of the file or record.

However, many corporations have huge databases, and having large numbers of users constantly accessing these data can, as we all know, create many more problems. The eastern region sales manager, let's say, needs only to look at a summary of the records for her own salespeople, but she is forced to sort through the records for the entire staff of several thousand to get the data she needs. Now, if it were possible to create an entity that would automatically access the data dictionary, and reference the database, and would contain only the summarized information for eastern region sales staff, she would only have to go through a small file, and there would be a big savings in CPU time. The sales manager would get her reports a lot faster too.

Let's call this entity a subfile. It's a self-describing file - that is, it contains its own minidictionary - and it has another very valuable property: it is portable. The subfile can be passed like a football from one application program to another, from one computer system to another. Now the user has a portable data file that is consistent with the corporate database, because it was created automatically by the 4GL from that database. The subfile now becomes the integration key between the development center language and the information center tools.

There is, however, another need for integration at the information center level. This is to integrate with

Hewlett-Packard's own offerings in the office automation area (such as Deskmanager, DSG, Easy Draw, and VisiCalc). What this means is that in order to transfer information to these products, the information tools must offer support of the same file structure as the HP products. This includes figure file support for all the graphics and word-processing products, and DIF file support to integrate with spreadsheets and analysis tools residing either on the mini or on micros. Figure file support also ensures compatibility with laser printers.



To summarize, information center tools should have a way to integrate both with the 4GL and with the office automation products widely used within the corporation.

Some Real-Life Situations

Imagine this situation. You're the product manager of a manufacturing division in a large corporation. As you sip on your first coffee of the morning and review a crowded schedule, the phone rings. It's the president calling to tell you he's accepted an invitation to speak at a major trade show, and he's just remembered that he needs needs a presentation. Nothing much, just a half-hour talk with lots of facts and figures, all the latest production projections, maybe ten or fifteen charts that he can use as slides. And by the way, he'll need that by the end of the day tomorrow.

After promising yourself that you'll never do this to your managers when you're president, you take a deep breath and begin. You have a desktop micro with a built-in terminal emulation, so you can log on to the HP3000. There you can use a simple report writer to extract some of the data you need.

You can also write this data to a financial planning application program via a subfile. Next, you consolidate several sets of financial data into a single report that meets your needs. Now combine the two reports, load them into a single subfile that can be converted to DIF format and downloaded to your micro. Copy the DIF file into your word-processor, and you're ready to begin preparing that half-hour presentation by putting some words together with the facts and figures. Later you'll load some of the figures back to the mini to make use of another application that will give you presentation-quality graphics.

Note that none of the processes involved here required the direct involvement of the MIS department. Thus the ripple effect of this sort of rush assignment was avoided - nobody else had to be inconvenienced to meet the president's need for instant data. Yet the user not only accessed those sections of the database that were available to him, he also manipulated the data into the formats he needed, and was able to extract it, rework it in his micro environment, and finally use a subset of this data to produce the graphics he needed.

Let's look at another situation. You're the Vice-President, Finance, and you want to be able to do ad hoc summary reports from a large database. You also want to use the data to produce budgetary projections and to graph trends. Finally, you want the same data to be accessed by your secretary to integrate with your quarterly reports to the board.

Together with MIS, you indicate what data you will be using continually. MIS then prepares a subfile for your use, in a format compatible with your spreadsheet tool. You can now read the data in that subfile whenever you need it, produce ad hoc reports using a self-serve information retrieval tool, load data into your analysis tool, manipulate this data, save the results in another subfile, and graph it either by using an HP graphics tool or your 4GL's graphics option. Those graphs can be saved as figure files and passed on to your secretary's word-processor to be integrated as part of your quarterly report to the board.

The benefits of integration

These are just a couple of examples of how a fully integrated system can be made to work. It can be seen that, in addition to meeting the requirements specified earlier for access to information, increased productivity, and guaranteed data security, a system that integrates information center products around a development center 4GL offers numerous other benefits:

Optimization Integration lets different groups of users access the system in different ways using the same data. It means you are optimizing the use of the system without overloading it, and without overtaxing the MIS department.

Flexibility Users can "grow into" the system as they become more experienced, or as their needs become more complex. They can also stick with their favourite software products, yet make the results available to others who use different tools.

Effectiveness The MIS department takes on a much broader role as it spends less time on routine requests and fixes, and more on developing applications that meet users' needs and providing advice and consultation.

Economy We're all aware that whether you're in the competitive business world or in an institutional setting, time is money. In an integrated system there is simply less time wasted looking for or supplying information.

Information centers around 4GLs. The concept is not new, in fact when you think about it, it's simply an updated version of the old adage about using the right tools for the job. Only now our tools can change and adapt to meet the requirements and skills levels of the users. Above all, it's a concept that makes sense.

LETS SCOPE IT OUT ... FILE TRANSFER BEYOND THE SNEAKER-NET.

Birket Foster
M. B. Foster Associates Limited
P.O. Box 580
Chesterville, Ontario
Canada K0C 1H0

The problems with connecting a micro-computer to the HP-3000 are always challenging. The many options for the way a micro is hooked to an HP3000 are as varied as the uses for the micros. In the author's travels to many user sites and regional users group meetings several problem areas become evident during discussions on this topic. The author will explore the use of Reflection - a third party product which runs on a wide range of MS-DOS micros - to transfer files. The use of REFLECTION in conjunction with the micro to create intelligent workstations will also be discussed.

The basic technical problem is that the terminal is not always connected by wires directly to the HP3000. The basic categories that interfere with the HP3000 and the micro talking easily include Modems, Data PBXs/switches, Phone switches using Data over voice, Local Area Networks and X.25 networks. This paper will explore the use of micros over these media.

As usual nothing in technology ever stands still. This paper will discuss kind of things to look for if you are intending to get out onto the bleeding edge of technology.

The talk will consist of:

1. The philosophy of the intelligent workstation
2. Some applications of file transfer.
3. How looking at the problem can save you a lot of money.
4. The micro - the connection - the HP3000, parameter settings and tricks to move through different media.
5. The future - what else should we plan for?

1. The philosophy of the intelligent workstation.

The evolution of input devices in the world of computers has influenced the progress of computing. It is hard to imagine vacuum tube computers and how you would communicate with and program them. Later paper tapes and punch cards became input (and output) media. The legacy of the punch card is still present even in your terminals of today. You can see the influence of the Hollerith card or punch card every time you work on a terminal with only 80 columns.

What could be beyond the right margin? How about reports which need 132 columns. How about the 24 rows on your screen? Well HP started terminals which used memory to scroll vertically. There was usually 3 or 4 pages of scrollable memory. Then came the HP150 delivered with 176,000 lines of micro-code, all kinds of memory, but only 2 pages of scrollable memory.

HP terminals have all kinds of neat features, HPWORD terminals which allow HPWORD stuff to be downloaded from the HP3000 (not to be done over a modem as it takes a while at the best of times), and Block mode, which allows data entry people to fill in a form. You can get forms caching to allow the form to be loaded from terminal memory rather than repainting it across the data comm line.

Then there are graphics terminals which allow you to preview the graph that you want to plot. Plot time is slower on the plotter than on the screen. Colour terminals add the ability to plot colour graphs on the screen.

The REFLECTION product from Walker Richer and Quinn allows the user of micros to put some intelligence into their workstations. PC2622 as REFLECTION was formerly named was the first software product to emulate HP terminals including block mode and ENQ/ACK handshaking.

An IBM/PC or HP150 with 640K running Reflection could expect 3,000 lines of vertically scrollable memory. The problem with the right margin has disappeared ... there are up to 9,999 columns of sideways scrollable memory. The command language enables a user to have the PC perform a set of instructions, even at the press of one keystroke.

Other features of Reflection are file transfer and peripheral management. You can send binary (like LOTUS WKS files) to the HP3000 as well as ASCII plain text files. You can print a session to disc or to printer.

The latest version of REFLECTION emulates a HP2397 colour graphics terminal and allows a background partition to be running at the same time. This will allow running as a terminal and then at the push of a pair of keys, an MS-DOS session can be started. Then you could bring up LOTUS and without leaving LOTUS flip to the terminal session check on your terminal session. Now if they could only support a phone in my computer, a laser disk for 10 gigabytes of storage and ...

2. Some applications of file transfer.

The primary form of file transfer in many organizations is the sneaker-net. That's when, to transfer a file to Joe, you put on your sneakers, grab your diskette and trot over to Joe's workstation.

Sooner or later you will run into a situation where the sneaker-net breaks down, a short jog is OK but how many of us are into marathons. At this point an alternate form of file transfer is necessary. The fact is that there is in the terms of economists a time value of information. The longer it takes the less valuable the information will be when it arrives. There will be situations where sending the diskette by federal express or mail will be satisfactory but there are bound to be occasions where online file transfer is the best way.

There are many ways of using file transfer. In our company we use file transfer to load new versions of software on customers machines. Instead of having to wait days for a new version it is available in hours.

In fact, when we work with our suppliers we can dial in and pick up new software long before the mail would have got it to us.

Several of our customers have been using micros to compile COBOL programs to clean out syntax errors without putting a burden on their HP3000. Basically they download the source code, edit and compile on the micro. Once the code compiles clean, a file transfer sends the source back to the HP3000 and it is available for testing.

There is a disease spreading through corporate America which is called the \$40,000.00 data entry clerk syndrome. This disease is easily spotted and can be cured permanently in about 2 days. Meanwhile in many organizations it wastes 3 or 4 mandays per month.

The way to spot the disease is to look for a manager who takes the figures off a report (usually the month end report) and rekeys them into a LOTUS spreadsheet. Now it seems to me that with all the technology we have that produces the report in the first place we should be able to download the information directly into the micro and move it into the LOTUS spreadsheet, without too much difficulty.

In fact if we are really lazy, there are several packages on the market which build files from your databases which are formatted for importing directly into LOTUS. If you want to learn how to do this for yourself there is an Appendix in the REFLECTION manual or a longer explanation by one of our customers, Tom Kaminski of Singer Career Systems in Rochester. Tom presented a paper in Washington which can be found in the proceedings of the Washington Interex conference.

3. How looking at the problem can save you a lot of money.

By exploring the ways by which a problem can be solved some times a significant amount can be saved. Here are some examples of what a second look at an approach can do.

In order to help get upper management support for Electronic mail (in yuppie thats E-Mail), you install terminals for your 8 members of the mahogany row management group. Even if they did know how to keyboard (thats a safe way to talk about typing in front of management), the problem is that they don't want to spend their time or tie up their secretaries' time dialing up the E-mail to see if they have any messages.

One solution is to write a command file which dials the electronic mail system logs on as each of the 8 members of the management team and copies any mail it finds to the disks on the micro. A couple extra lines of code and this will happen every 2 hours all through the day. A small menu driven system will allow the selection of the mail for printing. This is all possible with REFLECTION now.

Now all you need is some artificial intelligence to determine how to respond to each piece and then where to file it. The additional peripheral which will enable the non-power user to use this easily would be a Voxtrak system which literally reads the screen outloud. Of course a bio-monitor to check that the level of brain activity was conducive to listening and comprehending might also be desirable. Perhaps an executive mail pack from HP or a third-party will appear on the market to do all of the above while you go off fishing.

Then of course there are those of us who have to talk to the corporate big blue machine. Most of the time the 3270 emulation is used rather than get two terminals on a guys desk. One of the most popular ways of accomplishing this is to wire your floor with coax cables and use IRMA boards in PC's to allow the PC to pretend its a 3270 terminal.

Most organizations that are large enough to have that big blue machine also have either a data switch or are using their PBX for switching terminals. It turns out that there are several manufacturers of protocol convertors which hang off the back of these switches. Generally these take VT100 in from the PC end and feed back 3270. Even split screen SPF is possible.

The protocol convertors are capable of channel attaching to the IBM host. They also could come in from a remote concentrator. The advantage is that you can use existing wiring (assuming you have a phone at every desk or at least a terminal) to hook to the switch and even have dialup 3270 sessions.

This method is extremely cost effective for those places where there are users who need to be able to access both machines. Since many users are not logged on all day it makes best possible use of the ports on both systems. The HP and the VT100 terminal emulation are in the Reflection software package.

It turns out a company with 30-50 users, wanting to access the IBM host as well as the HP3000 can save \$60 - \$70 thousand using this scenario. All this just for thinking for a few moments.

4. The micro - the connection - the HP3000, parameter settings and tricks to move through different media.

If you are direct connected with a baud rate of 9600 you can start using REFLECTION right out of the box, no changes no fuss, muss, or bother. But now what about the rest of us?

Yes, files can be transferred, even binary ones across multiplexors (muxes), X.25 networks and other strange networks. A little knowledge about communications is required but not a whole lot. 99% of what is needed is in the REFLECTION manual ... most people just never get around to reading it.

The first and biggest mistake is changing the host prompt character on terminal page 2 from the left hand pointing arrow < to the colon : . This is guarenteed to mess up file transfers even under the best of circumstances. On an HP150 or Vectra or device with a large number of pixels on the screen this will be a DC1 character.

To deal with a multiplexor or LAN if the left hand arrow (its really a DC1) doesn't show when display functions is on, then change the host start-up sequence on the config. xfer screen to read:

```
RUN PCLINK.PUB.SYS;PARAM=1
```


When an X.25 network is used, log on as termtyp 24 and make these changes.

| | | |
|-------------------|----------------|-----------------|
| Check parity | to No | (comm port) |
| receive pacing | to XON/XOFF | (comm port) |
| transmit pacing | to XON/XOFF | (comm port) |
| enq/ack pacing | to NO | (comm port) |
| Inhibit Handshake | to YES | (terminal pg 1) |
| Inhibit DC2 | to YES | (terminal pg 1) |
| Host prompt | to NONE (CTL@) | (terminal pg 2) |

There are several other areas that can slow you down. For instance X.25 networks which don't have both PADS configured correctly can significantly increase your traffic on the network.

Remember that the first time Reflection is used for file transfer the micro must be connected directly into the HP3000 or have an eight-bit data path because it is required to upload PCLINK. Preferably you will be logged on as the System Manager so that all the users on your system can use the same copy without having to each modify their config. xfer screen.

5. The future - what else should we plan for?

There are lots of plans for future communications products which will affect the way that you do business. Some of the areas which are still working on standards are:

PBX to Mainframe (Read 930's) communications. The short term 2-3 year solution is probably going to be CPI. The basic concept is a backplane to backplane communications. Northern Telecom is a major promotor of this technology. The other PBX-Mainframe solution is DMI. This is the standard being supported by A T & T. Rumour has it that HP has been testing the DMI interface at 9600 and 19.2 in the labs, hopefully thats on the 70 not the 930. This appears to be the 3-5 year time frame solution.

LANs for the office are getting a lot of vendor press as each will solve all the problems of automating the office. The WRQ labs are already supporting the HP Office share LAN, the Novell network and working on several others. Seems it will be 18 months before the LAN will become an easily installable and supportable reality.

LANs for the HP3000 are a hot topics. By using IEEE 802.3 (read Ethernet) as a standard, HP opened more bandwidth between machines. Already several clients have been pleased by the increased throughput possible. There are several of our clients looking at interfacing the LAN/3000 to DEC/VAX.

The DTCs are a nice concept. Anyone who has ever pulled cable will appreciate the idea of being able to run one cable to the accounting department rather than 40. There are already people who used HPs fiber optics to solve that problem and kept conventional RSC232 communications. The folks at Compaq Corp have a fellow who has figured out how to provide shared resources and connect up to 1000 devices to an HP3000. I think he's using x.25 and the virtual session capability of LAN/3000.

If you are thinking of buying a switchboard in your company there are good reasons for the Information Systems group, Data processing department or what-have-you to get involved. There is just too much coming down for the administrative services group to be allowed to purchase it all on its own. If you aren't familiar with data over voice, modem pools and virtual terminals you better do some reading or you'll get some real surprises. The next set of terminals will be intelligent work stations with phones built into them. These will all have the option of handling ISDN networks, a "soon to arrive" networking standard for digital data networks.

All the mergers between computer companies and PBX companies have not been by accident. It looks like the long term survival will depend on knowing how to make computers work with the soon to be available networks. Where is HP's switchboard company?

The term "smart building" has been used for several years to describe buildings already wired for the next generation of PBX's. But since better than 80% of the buildings which will be standing in 2000 are currently built you can bet that the BOC's and communication companies are going to continue to try to stretch the capacity of existing copper wire. Reports currently show 1 megabit per second and testing is going on at the 10 megabit per second level.

The importance of communications is going to grow over the next few years. New equipment, new offerings from the BOCs and new protocols will make your job even more challenging. Who knows maybe HP will equip its sales force with Portable Pluses and cellular telephones to remind its customers that it is doing some leading edge work. The phones could come from several different HP customers. After all even federal express has portable terminals for its delivery people.

File transfer at that point will be from well beyond the sneaker-net.

Birket Foster is the President and founder of M.B. Foster Associates Limited, a nine year old software distributor with clients including most of North America's largest companies. Birket is very involved in user group activities, his company belongs to 12 regional users groups. Birket gives presentations at user group meetings on a regional, national and international level. In his spare time Birket is the chairman of the Special Interest Group for Software Vendors (SIG-SOFTVEND). Birket met his first HP3000 in 1974 ... it was love at first byte.

TRANSACT
ADVANCED PROGRAMMING TECHNIQUES

Stephen M. Butler
PROBUS INTERNATIONAL, INC.
8815 - 106th St. E.
Puyallup, WA 98373

Jim McIntosh
McIntosh and Associates
621 - 1st Ave. West, Suite 306
Seattle, WA 98119

The language now called TRANSACT has been around since the late 70's. Yet, there persists a major lack of knowledge regarding what is good versus what is bad programming practices. Most of the current courses teach the syntax without indoctrinating the student in the approach to proper usage. The authors have seen and worked on many TRANSACT programs that are very reminiscent of COBOL. The verbs are different, but the structure is very COBOLish. This paper is neither a tutorial nor a solution generator. Our purpose is to raise points of discussion (more in depth at the conference on those points of common interest) that the reader can research to find the appropriate method of approach for a given situation. First, some general information about TRANSACT (TPL) and an outline of our basic premise.

TPL is NOT a 4th generation language. A 4th generation language is simply told what the end result is to look like and with what information the result is to be obtained. The algorithm or procedural method is left up to the language to determine. This is the case with REPORT and, to a greater extent, INFORM. With TPL, the programmer must specify the procedural method in a manner similar to COBOL, BASIC, FORTRAN, etc. The main difference between TPL and other 3rd generation languages is in the high level interface to IMAGE and VPLUS. It is these high level verbs (or Multi-operational verbs) that distinguish TPL from the other procedural languages. These same verbs are the greatest stumbling block to proper optimization of TPL. Along these lines we take the motto coined by Larry Kemp of HP, Bellevue, WA--"Less is More".

THE STACK

How? The most obvious is reduction in stack size. We all know that smaller stacks will fit inside the HP much easier than larger ones. Additionally, since the TPL processor is in reality an INTERPRETER reducing the PCODE size reduces the number of instructions that need to be interpreted. Therefore, using the high-level verb constructs (eg FIND(CHAIN); vs. REPEAT DO...GET(CHAIN) ,STATUS;...DOEND;) gives a double benefit (reduced stack size and less interpreter overhead).

Other methods for reducing the stack size are:

1. Use OPTI to remove unused items from the internal program storage.

2. Along with #1, use OPT on any item whose name is not needed by the program (eg, I/O to IMAGE). TPL keeps the names in the stack otherwise. If the name must be in the stack, then give it as short a name as possible. Note that if you have defined an item to be a synonym of another, only the name actually used by IMAGE needs to be kept. The other name can have OPT put on it's definition.
3. DEFINE or reference very early in the code the most frequently used items within the program (by count--not algorithm). The first 255 items are referenced via a single byte allowing most TPL pseudo code to be one word in length. The other items require a full word to reference them and expand the pseudo code instruction to a double word at best.
4. Maximize the use of common literals. If a number of literals make use of common strings (of more than 10 characters in common) then break up the literals so that the common part is enclosed in quotes. When the compiler finds the following:

```
"PAGE " "1"
```

it will store 'PAGE ' separate from '1' but will still treat it as 'PAGE 1' for purposes of a DISPLAY, MOVE or any other place a literal is allowed (eg, WINDOW="..."). Any other use of "PAGE " (as in "PAGE " "2") will only store the '2' as the 'PAGE ' is already stored. Remember, it takes about 10 characters in common to make this really cost effective.

5. For messages or other displayed literals that are rarely used (as in error messages) consider the use of a message catalog. In addition to reducing the stack size, it allows you to change the messages without recompiling the program (eg, spelling mistakes such as this one). The trade-off lies in the need for another file to be opened (on the SYSTEM statement and an explicit FILE(OPEN) statement) and I/O to obtain a particular message.
6. For VPLUS applications two items are very critical:
 - A. Explicitly state in the SYSTEM statement the forms that are to be used from the forms file. Otherwise, the compiler will use stack space to define all the forms and all the items found in DICTIONARY for this forms file.
 - B. Use FAST version of the forms file.
7. Properly segment the TPL code (if you decide to segment). Other aspects of segmenting are discussed later; but, the

common rules from the other languages apply also. Make the segments all the same size and don't branch from segment to segment indiscriminately. With TPL, the root segment is always in the stack (and therefore should be as small as practical and contain common routines) with the other segments being put into the overlay area. Thus, the largest overlay segment determines the amount of stack space carved out.

THE PCODE

Following the "Less is More" theme, the programmer is encouraged to locate and utilize the least amount of code needed to get the job done. This includes locating those high powered data access verbs and use the options very sparingly.

The MULTI-OPERATIONAL data access verbs provide the user with inherent loop constructs without coding them. For example, a FIND(CHAIN) with a PERFORM=option will operate much faster than coding a loop with a GET(CHAIN) (and needing to use the STATUS option). The TPL processor must decode the FIND(CHAIN) only once for the duration of the chained read(s). The other method requires interpreter overhead to follow the coded loop. One of the authors (Butler) took a program written in the best form of COBOL (yes, that is a snub!) and applied that simple rule. The program used to take 24 hours. After the rewrite, 90 minutes! (The sad part is that a FASTRAN run of both programs gave the rewrite only 4 minutes advantage over the inefficient one--36 to 40 minutes).

The less options you need to use the faster the program will run. There are less control words to interpret and the automatic error handling normally defaults to what needs to be done anyway. If automatic error handling isn't correct for your application, then use the ERROR=label option rather than the STATUS option. This will allow the multi-operational verbs to keep their multi-operational state. The STATUS option forces the programmer to code a lot of extra verbs that are done automatically otherwise. Along with the ERROR=label option, consider whether the standard error message is to be displayed--if not, then include the NOMSG option also.

Another option that sounds nice at first--but opens many cans of worms--is the SORT= option. Especially nice on the FIND(CHAIN or SERIAL)--also especially deadly. First, all the records are read into another file to be sorted (FASTRAN uses an input procedure instead) and then SORT grabs all the stack space available--oops, there goes that stack again! When the records are finally made available to the code, the current record from the database is no longer available (unless you used the RECNO=option). Therefore, doing an UPDATE in the middle of a PERFORM= option when the SORT= option is also used is an invitation to disaster. This option is especially sinister in that it is so easy to include on any of the multi-operational data access verbs (FILES and DATASETS--not VPLUS). If you must use it, KNOW what is happening--don't assume that TRANSACT will keep track for you. It doesn't!

ARITHMETIC PERFORMANCE

There are several items in addition to keeping the PCODE small and tight that will add to the performance.

1. In arithmetic operations--combine them up into one statement as opposed to a whole string of LET verbs. This is one case where "More is Less" and one should be on the look out for places to combine two or more LET statements together.
2. Use literals sparingly within the LET statement. If you repeatedly must loop through a construct of:

```
LET (COUNT) = (COUNT) + 1;
```

then define an item called ONE and initialize it to 1. For ever afterward use it in place of the literal.

3. Don't mix data types. If you can keep everything as a single word integer (J(4) or less), then the processor will do integer arithmetic. Otherwise, everything gets converted to packed format (P format) for the operation and the result is converted back. These conversions are very expensive.
4. If possible, store you data in P format in your files. This will eliminate the need to convert to and from packed format for arithmetic operations.
5. Don't use the LET statement to initialize a variable to zero. For non-display numerics (P,J,I,R),

```
MOVE (var) = "";
```

This will move LOW-VALUES (or binary zeros) to the field in question (also a good trick for initializing a string or other big chunk of data). If the numeric is of display type (9,Z) then MOVE (var) = "0000" with as many zeros as needed for the internal byte size.

6. To simply copy a variable to another then use the MOVE statement. Be careful here that both the destination and the source are defined EXACTLY alike--even to the number of decimals. Now, if you KNOW what is happening you can use the MOVE trick to multiply by some power of 10. This is mainly useful on integer or packed types where only the decimal part changes. The internal size must ALWAYS be the same.
7. Remember that the LET OFFSET(var) = is one of the most time consuming statements in TPL. Reduce the usage of this to a minimum. There are other techniques involving

TRANSACTION: ADVANCED PROGRAMMING TECHNIQUES

LIST manipulation that are better suited for many array type applications. But, if you must use the index type approach with the OFFSET statement, then group as many items as you can together as a child to the item you OFFSET through the array. This in effect moves all the children without doing an OFFSET on each one. Note that this is indeed an OFFSET and not an OCCURENCE! OFFSET starts at 0.

LIST MANIPULATION

Judicious use of the LIST, and SET(STACK) verbs will buy many dollars worth of CPU. The first rule is to keep your LIST register as short as practical. Don't put everything on up front and keep it there. Treat this as a real, live, honest-to-goodness, stack! It is much faster to pop items off the stack and remap the DATA register with new item names than to utilize the MOVE statement to copy the data. The one caveat to this is--don't manipulate the stack while inside the PERFORM= option of a multi-operational verb. But this too can be a blessing in disguise. For example, a FIND(CHAIN) with a LIST=(a:b) option will normally pass the item list to IMAGE just once. On subsequent records the '*' is passed and TPL stuffs the record back into the same spot in the data register. HOWEVER, if the list is manipulated during the perform then TPL notices and assumes that you may have a different set of items to be picked up. This is a great way to build up an array by just adding the same items to the list inside the PERFORM paragraph. Note also that if you use marker items, you could change the items you were picking up. Say you wanted item A from the first record, B and C from the second and D,E,and F from all the others. Now, as those old textbooks used to say, the implementation is left to the reader.

While earlier we stated that you should define or reference the most used items early in your program so that they fall within the first 254 items used, we now add that the immediately localized items (ie, those items within the same code location) should be the last items added to the stack. TPL must search back from the last item put on the stack to resolve references. Thus the need to keep the list fairly short and to manage it well.

One comment regarding SET(STACK) LIST(item) and SET(STACK) LIST(*). The latter will POP the last item added off of the stack. The former is more subtle. If the stack (from bottom to top) contains: C,D,B,A,Y,J,A; a SET(STACK) LIST(A) will remove the last three items from the stack. The search for a SET(STACK) starts one below the top. If the item is not found (and only then), the top is checked. So, if all you want to do is POP the last guy off; do it via LIST(*) rather than LIST(item).

As far as possible, ensure that VPLS item names and IMAGE item names are in fact the same. TPL will take charge of doing any data conversion necessary. The added benefits are:

1. You don't have to do the numeric conversions.

2. No MOVE or LET statements are necessary.
3. No LIST manipulation is required.

WHEN LESS IS TOO MUCH

There comes a time in the life of every program (MURPHY'S LAW OF UNIVERSAL EXPANSION) that it is too big to fit inside the constraints of the HP-3000 and more explicitly, inside the stack that TPL manages. There are three distinct ways to break the code up. Each has advantages and disadvantages. They are: PROC'ing, SEGMENT'ing, and CALL'ing. We'll look at them in that order.

The PROC statement is used to make calls to SL routines. These will have been written in another language to do a specific job. The main items to remember are:

1. Be sure to DEFINE(INTRINSIC) anything that TPL will accept in that category. This may change (ie be added to) from release to release. This essentially tells the processor that no LOADPROC is needed. The processors are already linked and they are available by virtue of the processing being run by you.
2. Those PROCs that are rarely used (eg, call to GETMESSAGE for message catalogs) should have the NOLOAD option. If after it has been called, you are fairly certain the program will keep on running without making another call, then specify the UNLOAD option. But be careful. Don't unload it just to turn around and load it back up again.
3. If your programs use the CALL statement (see below), and both the caller and the callee will use the same SL segment, ensure that the caller makes the first call. That way, the procedure will get loaded for the entire time the caller is running. Otherwise, the callee will do the loadproc and upon exit, unload it. Now, that is called cleaning up after oneself! But, not in our best interest!
4. If the subroutine is written in COBOL, be sure to include the CLANG(COBOL) parameter in the PROC statement. This will invoke the COBOL traps so that the TPL processor will not see any of the "normal" errors that COBOL generally runs into. NOTE: If you are calling COBOL from something like SPL or FORTRAN, you should get the COBOL LIBRARY TRAP set up in advance of the call. Optionally, you can have the subroutine do that as its very first item of business.

SEGMENT'ing has certain advantages in TPL. For one, there can be 127 overlay segments (plus the root segment). That can lead to a lot of

code! But, when the program is segmented, the PCODE file is kept open for the duration. So,

1. Be sure there are two overlay segments. If there is just the root and one overlay segment, then nothing is accomplished and in fact, more stack space is used than needed.
2. Keep the size of the overlay segments fairly constant. The stack space used will be according to the largest overlay.
3. Put the common routines and global items into the root. Two or more overlay segments use an item, it should be declared in the root so it will be global.
4. Switching from an overlay segment to the root and back to the same overlay is fairly trivial. It is very expensive to bring a new overlay in--so, stay in the overlay/root combination as long as possible (move code or duplicate code if needed) before changing to another overlay.
5. Use OPTS only after you have thoroughly debugged the program. Leaving local items on the stack intentionally may not do what you think is happening. Local items are numbered on up from where the root item numbers ended. Just because you have an item DATE defined as X(8) in segment 1 (and you leave it on the stack going to segment 2) and segment 2 has a local item called DATE defined as 9(8) doesn't mean that they will get the same item number. In fact, the item number used by DATE in segment 1 may very well define an item in segment 2 called NAME X(30). So much for local items on the stack tricks. They bite!
6. But, once debugged, do use OPTS to remove that checking.

The CALL statement as implemented by TPL allows multiple programmers to develop independent programs in parallel. The only requirement is that everybody understands what is being passed via the data register during the transition from one program to the next. It is up to the CALLEE to map that area with the LIST register.

1. This type of interface is slower than either PROC or SEGMENTS, but it is easier to develop, maintain, etc.
2. Declare the BASE and VPLS (and PROCEDURES) in the CALLER. Otherwise, the CALLEE will utilize the resources and then close them. If the CALLER does it first, they will remain open from one CALL to the NEXT. This is a very significant performance criteria.

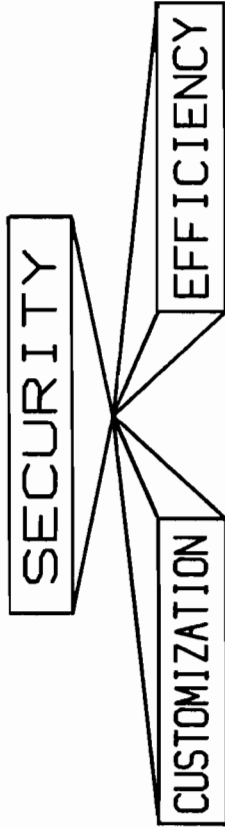


3. The DATA= area must be large enough in the CALLER to handle whatever the CALLEE needs (or anything the CALLEE may CALL).
4. Don't go too deep in CALLs. They all leave information on the stack. It is better to have a traffic cop.
5. Only the DATA register is shared among the programs. The other registers (ITEM, WORKSPACE, etc) are local to each program and should be managed as small as possible without undo hardship.

The CALL statement allows the programmer to reference an item name rather than a hard coded literal. This allows a generic type driver to be written that can act as a traffic cop between modules. We have found that using an approach like that works very well. The CALL level is kept to one level. Each module is for a specific task that the user is likely to remain in for some time. We utilize it with VPLUS so that a user presses a function key and the current program determines that it doesn't handle that function and exits to the driver with the number of the key that was pressed. The driver has a configuration file for each module which indicates what module is to be called based on the function key returned by the CALLEE. The drawback is in going to FASTRAN. Each of the called modules now becomes an SL entry and FASTRAN must do a LOADPROC. Currently FASTRAN doesn't remember what has previously been loaded. Therefore, there are some performance issues to be negotiated, balanced, etc.

Once more, "LESS IS MORE". Do less coding so the code is smaller, takes less time to code it, runs faster and is easier to maintain. By doing less work (once you've learned what LESS is), you become MORE productive.

"Enriching your POWERHOUSE environment"



Presented by: David G. Robinson

**R O B I N S O N,
W A L L A C E &
C O M P A N Y**
Software, Consulting and Training

"Your Full-Service, Value-Added Source for POWERHOUSE Accessories"



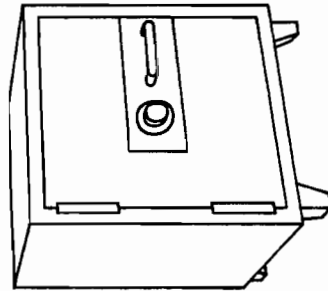
OVERVIEW

SECURITY

INSTRUCTOR'S CORNER

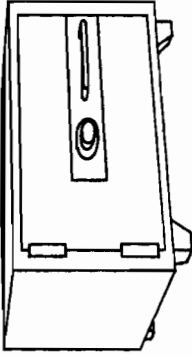
Is it possible ... RECORD SECURITY ?
TIPS & TECHNIQUES

- Password Security
 - Access by Port No.
 - UDC's
- SECURITY:** Other options





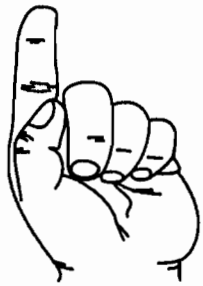
? RECORD SECURITY



SECURITY

I have an accounting file with combined data from several departments. Is it possible to restrict my users from seeing other dept records when using QUIZ? There is certain financial information that is privy only to the respective departments.

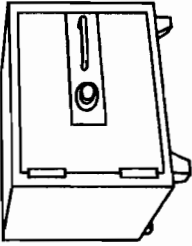
INSTRUCTORS CORNER



? RECORD SECURITY

Yes, assuming that there is a constant value in each record that explicitly identifies which department it belongs to.

Enriching your POWERHOUSE environment



SECURITY

```

: QUIT
> ACCESS Acctg
> REPORT Dept-Name &&&
> > Dept-Code
> > Dept-Mgr
> > Dept-Budget
> GO
  
```

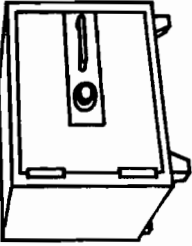


VP Finance ...

Jan/01/86 APPGEN INDUSTRIES, INC. Page 1

| Dept Name | Dept Code | Dept Manager | Dept Budget |
|-----------------|-----------|-----------------|--------------|
| Engineering | 01 | J.L Behrens | \$ 5,000,000 |
| Data Processing | 13 | Hard Luck Harry | \$ 15,000 |

Not Good!



SECURITY

... [QDD] Data Dictionary Security "OVERVIEW" ...

- Specify user classes with valid Logon IDs (SIGNON)

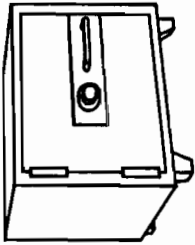
| | | | |
|------|---------|--------|----------------|
| USER | prodmgr | SIGNON | manager. sys & |
| USER | audit | SIGNON | manager. prod |
| USER | clerk | SIGNON | auditor. prod |

| | | | |
|------|-------|--------|--------------|
| USER | audit | SIGNON | bill. prod & |
| USER | clerk | SIGNON | sue. prod |

- Implement READ/WRITE INCLUDE/EXCLUDE user class/UNKNOWN/ALL at FILE or ELEMENT level

| | |
|--------------|-----------------|
| FILE payroll | TYPE KSAM & |
| READ | EXCLUDE ALL & |
| WRITE | INCLUDE prodmgr |

- * WRITE access implies READ access
- * ELEMENT security requires READ access at FILE level



SECURITY


 Potential Assassination attempt on VP ...
 or 4GL's Strike Again ...

SOLUTION

```

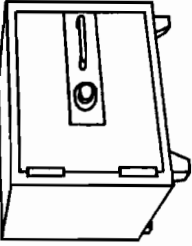
USER      mgr      SIGNON      vp. finance
USER      class1  SIGNON      mgr. dp
USER      class2  SIGNON      mgr. engr
FILE      acctg    TYPE MPE &
READ EXCLUDE ALL WRITE INCLUDE mgr
FILE      dp-recs  TYPE MPE &
OPEN acctg READ INCLUDE class1, mgr
FILE      engr-recs TYPE MPE &
OPEN acctg READ INCLUDE class2, mgr
RECORD   acctg
RECORD   dp-recs
ITEM     dept-code
RECORD   engr-recs
ITEM     dept-code
    
```



RECORD SECURITY

SELECT "13"

SELECT "01"



SECURITY

PASSWORD SECURITY

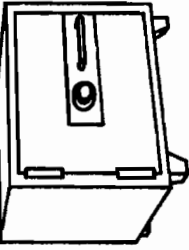
- Require passwords to ACCESS specific screens or functions
- Passwords kept in external file; Easy to maintain
- QUICK DESIGNER PROCEDURE

PORT SECURITY

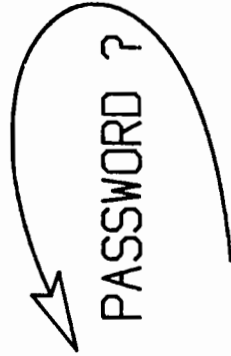
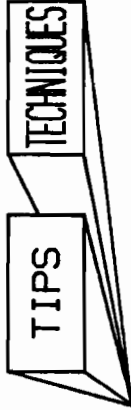
- Restrict access to certain terminals [PORTs]
- QUICK DESIGNER PROCEDURE and FUNCTIONS

UDCs

- LOGON UDCs restrict the USER to specific Application(s)
- Enhance LOGON UDCs to allow STREAMING of JOBS



SECURITY



PASSWORD SECURITY ?

MODE - ACTION _____

APPGEN INDUSTRIES, INC

01 Detail Listing of Files

02 ACCTG Data Base



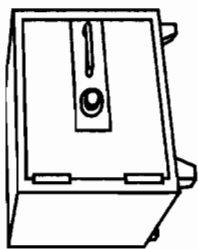
PASSWORD SECURITY !

```

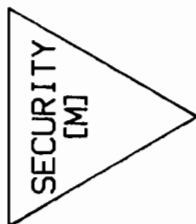
SCREEN ... MENU
FILE security DESIGNER
TEMPORARY password STRING #8
TITLE "APGEN INDUSTRIES, INC." CENTERED
COMMAND "LISTF,2" LABEL "Detail Listing of Files"
FIELD password LABEL "ACCTG Data Base " NOECHO

PROCEDURE DESIGNER 02
BEGIN
ACCEPT password
GET security using ("ACCTGDB " + password) OPTIONAL
IF ACCESSOK
THEN RUN SCREEN scacctg CLEAR SCREEN
ELSE ....

```



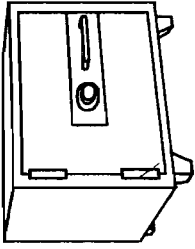
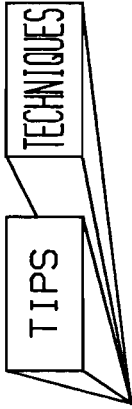
SECURITY



```

RECORD SECURITY
ITEM full-key CHAR SIZE 16 &
UNIQUE KEY
REDEFINED BY
ITEM resource CHAR SIZE 8
ITEM password CHAR SIZE 8
END

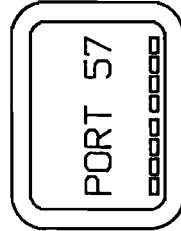
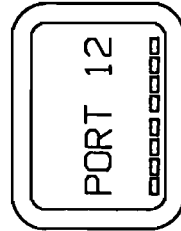
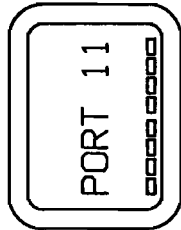
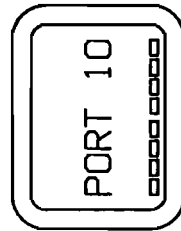
```

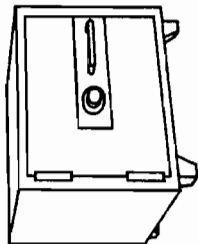



SECURITY

PORT SECURITY ?

Restrict applications to SPECIFIC Terminals [PORTNUMBER] ?





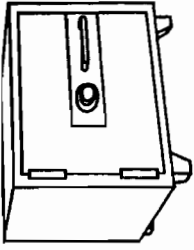
SECURITY

PORT & PASSWORD SECURITY !

```

PROCEDURE DESIGNER 2
BEGIN
  IF 0 LT INDEX ("10 + 11 + 12 + 57", ASCII(PORTNUMBER))
  THEN BEGIN
    ACCEPT password
    GET security USING ("ACCTGDB " + password) OPTIONAL
    IF ACCESSOK
    THEN RUN SCREEN scacctg CLEAR SCREEN
    ELSE ERROR " Invalid Password ##### "
    END
  ELSE ERROR " ***** Unauthorized Terminal ***** "
END
BUILD

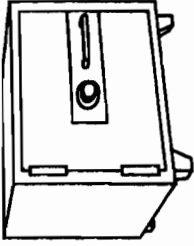
```



SECURITY

UDC's ?

- . UDC's make start-up convenient; take operator directly into first screen, by FILE equating QKGO
- . First screen can be Application-Specific or Top-Level Menu, with all choices below
- . Latter OPTION, with LOGON, NOBREAK User-Level UDC, is excellent security feature
- . Allow STREAMING of JOBS in this mode



SECURITY

UDC's

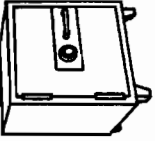
Example of POWERHOUSE UDC ...

```

Logon
OPTION LOGON, NOBREAK
SETJCW CIERROR = 0
CONTINUE
RESUME
IF CIERROR <> 978 THEN
  file equates =
  FILE QKGO = screen1
CONTINUE
QUICK
BYE
ENDIF

```

This UDC is derived in part from ideas contained in:
 "MPE Programming", by Eugene Volokh



SECURITY



SECURITY: Other Options

Physical Security

- . Computer Room
 - . Keep out what belongs out
 - . persons & things (e.g. destructive devices, floods)
- . Keep in what belongs in
 - . hardware & software
- . Big PROBLEM today is Data Communications
 - . secure modems (auto-call-back, etc.)
 - . encryption of message traffic
- * TERMPASS / SECURITY 3000 by VESOFI, Inc.
 - . sets passwords on dial up lines

Sign-on Security

- . USER, GROUP, ACCOUNT may be password protected
- . no passwords needed for home groups
- * SECURITY/3000 enhances sign-on security
 - . easier to remember personal data
 - . random choice of questions make it harder to "give away" your own password

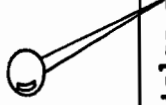
Data Structure Security

- . MPE & KSAM
 - . "barriers" at account, group, and file levels
 - . restrict access to file with LOCKWORD
- . IMAGE
 - . implement Image security
 - . privileged mode restricted to key personnel

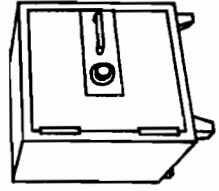
"Enriching your POWERHOUSE environment"



Remember SECURITY is an everlasting battle
 ... but you should always "endeavor to persevere"



- Articles on Security**
- " HP 3000 Security: A Practical Look", January, 1984
by Wayne Erfling, INTERACT
 - " Data Center Security", May, 1984
by Bruce Taback, INTERACT
 - " Security Penetrations", October, 1985
by Bruce Taback, INTERACT
 - " A Penetration Sampler", November, 1985
by Bruce Taback, INTERACT



SECURITY

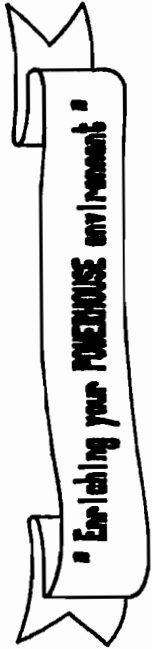


OVERVIEW



CUSTOMIZATION

INSTRUCTORs CORNER
Is it possible.. QUIZ Report Prompting ?
TIPS & TECHNIQUES
• Using DESIGNER Files for the End-User
• Custom Lookups for the End-User



? QUIZ Report Prompting

CUSTOMIZATION

I would like to have the ability from a QUICK Menu type screen to be prompted for a QUIZ "canned report". Is this possible and is it possible also to specify where the report's output is to be directed to ?

INSTRUCTORS CORNER



? QUIZ Report Prompting

Yes, by using temporaries and PROCESS Procedures this can be easily accomplished.



? QUIZ Report Prompting

CUSTOMIZATION

MODE - ACTION _____

APPGEN INDUSTRIES, INC

Order Entry Subsystem - Main Menu

- 01 APPGEN CUSTOMERS
- 02 SALES MASTER SCREEN
- 03 APPGEN CUSTOMER'S ORDERS
- 04 TRANSACTION SCREEN

10 QUIZ - REPORT ? _____
 Default Device [LP] _____



QUIZ Report Prompting is POSSIBLE !



CUSTOMIZATION



```

SCREEN okmenu MENU
TEMPORARY quiz-report          STRING #26
TEMPORARY quiz-rep-equate     STRING #39
TEMPORARY quiz-device         STRING #3
TEMPORARY quiz-dev-equate     STRING #21
SUBSCREEN ...
ALIGN (10, 13, 35)
FIELD quiz-report            ID 10 LABEL 'QUIZ - REPORT ?'
FIELD quiz-device           ID SAME LABEL 'Default Device [LP]' &
                           DEFAULT "LP"
COMMAND quiz-dev-equate     ID SAME NOLABEL
COMMAND quiz-rep-equate     ID SAME NOLABEL
PROGRAM mpex.pub.vesoft     INFO "%QUIZ" &
                           ID SAME NOLABEL
COMMAND "reset quizuse"     ID SAME NOLABEL
PROCEDURE PROCESS quiz-report
BEGIN
  LET quiz-rep-equate="FILE QUIZUSE=" + quiz-report
END
PROCEDURE PROCESS quiz-device
BEGIN
  LET quiz-dev-equate="FILE QUIZLIST;DEV=" + quiz-device
END
    
```



CUSTOMIZATION

? QUIZ Report Prompting

MODE - ACTION _____

APPGEN INDUSTRIES, INC

Order Entry Subsystem - Main Menu

01 APPGEN CUSTOMERS
 02 SALES MASTER SCREEN
 03 APPGEN CUSTOMER'S ORDERS
 04 TRANSACTION SCREEN

10 QUIZ - REPORT ? CUSTRPT3.PUB
 Default Device [LP] 183





CUSTOMIZATION

Using DESIGNER Files for the End-User

- . To automatically generate ORDER Numbers and maintain uniqueness
- . MPE file retains ORDER Number value as DESIGNER File in SCREEN
- . ENTRY PROCEDURE is modified to contain updating/locking logic

Custom Lookups for the End-User

- . To enable order entry clerks to search for parts by description
- . Automatically retrieve PART Number value by its desc and return to initial SCREEN
- . Usage of INPUT PROCEDURE and passing/receiving of temporaries



Customizing DATA ENTRY ! Using DESIGNER Files for the End-User



CUSTOMIZATION

MODE E ACTION U

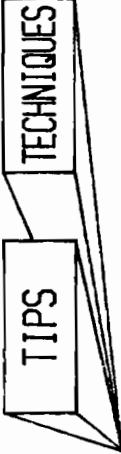
APPGEN INDUSTRIES, INC

| | | | |
|----|--------------|--------|--------------------------|
| 01 | Order Number | 1001 | <input type="checkbox"/> |
| 02 | Cust ID | C001 | <input type="checkbox"/> |
| 03 | Date Placed | 010186 | <input type="checkbox"/> |
| 04 | Sales Clerk | SC01 | <input type="checkbox"/> |

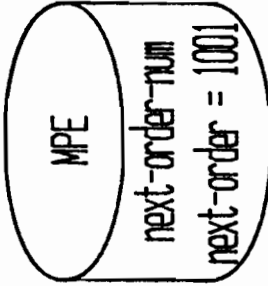
MODE E ACTION —

APPGEN INDUSTRIES, INC

| | | | |
|----|--------------|------|--------------------------|
| 01 | Order Number | 1002 | <input type="checkbox"/> |
| 02 | Cust ID | — | <input type="checkbox"/> |
| 03 | Date Placed | — | <input type="checkbox"/> |
| 04 | Sales Clerk | — | <input type="checkbox"/> |



CUSTOMIZATION



Customizing DATA ENTRY ! Using DESIGNER Files for the End-User

```

SCREEN qkohdr
FILE order-header PRIMARY
FILE next-order-num DESIGNER
FIELD .....
PROCEDURE ENTRY
BEGIN
  LOCK next-order-num
  GET next-order-num USING 0
  LET order-number=next-order
  LET next-order=next-order + 1
  PUT next-order-num
  DISPLAY order-number
  UNLOCK
  ACCEPT ...
END
RECORD next-order-num ZONED
ITEM next-order

```



Customizing DATA ENTRY ! Custom Lookups for the End-User



NAME & ACTION _____

| ORDER LINE ITEMS | | | | |
|------------------|-------|-----------------|---------|-------------|
| Line# | Part# | Description | Qty Req | Extension |
| 01 | 1 | CA000001 | | |
| 02 | 2 | Ink Jet Printer | 10 | \$ 4,950.00 |
| 03 | 3 | NY000050 | 5 | \$ 1,875.00 |
| 04 | | LOOKUP | | |
| 05 | | | | |

Enter "LOOKUP" for DESC SEARCH or "Invalid Part No."

TECHNIQUES

TIPS

Enriching your POWERHOUSE environment



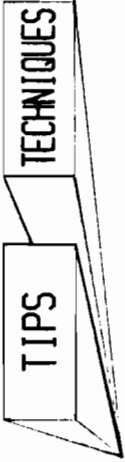
CUSTOMIZATION

Customizing DATA ENTRY ! Custom Lookups for the End-User

MODE 8 ACTION 3

"DESC SEARCH" PART NUMBER

| PART# | Description | Flag |
|-------------|-------------------|------|
| 01 CA000001 | Ink Jet Printer | |
| 02 CA000002 | Plotter Pens | * |
| 03 CA000003 | Printer Paper | |
| 04 NY000050 | Intercom Modem | |
| 05 NY000099 | Lasat Jet Printer | |



Customizing DATA ENTRY ! Custom Lookups for the End-User



NAME & ACTION _____

| ORDER LINE ITEMS | | | | | |
|------------------|----------|-----------------|---------|-------------|--|
| Line# | Part# | Description | Qty Req | Extension | |
| 01 | CA000001 | Ink Jet Printer | 10 | \$ 4,950.00 | |
| 02 | NY000050 | Internal Modem | 5 | \$ 1,875.00 | |
| 03 | CA000003 | Printer Paper | | | |
| 04 | | | | | |
| 05 | | | | | |

Enter "LOOKUP" for DESC SEARCH or "Invalid Part No."

Enriching your POWERBASE environment

TIPS

TECHNIQUES

Solution for Custom Lookup "DESC SEARCH" ! CUSTOMIZATION



Data Entry "Line-items" SCREEN

```
SCREEN qlitem
FILE line-items PRIMARY OCCURS 5
FILE parts-master REFERENCE
TEMPORARY flag-part          STRING *8
```

```
..FIELD part-number OF line-items   REQUIRED NOCHANGE &
LOOKUP ON parts-master &
MESSAGE 'Enter "LOOKUP" for DESC SEARCH or "Invalid Part No. "'

..PROCEDURE INPUT part-number
BEGIN
IF FIELDTEXT = "LOOKUP"
THEN BEGIN
LET flag-part = " "
RUN SCREEN qkfpart MODE S PASSING flag-part
LET FIELDTEXT = flag-part
END
END
BUILD
```

TECHNIQUES

TIPS


 Enriching your PAPERHOUSE environment

Solution for Custom Lookup "DESC SEARCH" !

CUSTOMIZATION



Inquiry "DESC SEARCH" SCREEN

```
SCREEN qkfpart ACTIVITIES FIND RECEIVING flag-part
TEMPORARY flag-part STRING #8
FILE parts-master PRIMARY OCCURS 5
TEMPORARY select-flag STRING #1 OCCURS WITH parts-master
```

```
::FIELD select-flag ID SAME
```

```
PROCEDURE DESIGNER 01
```

```
BEGIN
```

```
FOR parts-master
```

```
BEGIN
```

```
LET select-flag = "
```

```
DISPLAY select-flag
```

```
END
```

```
LET select-flag = "*"
```

```
DISPLAY select-flag
```

```
LET flag-part = part-number
```

```
END
```

```
BUILD
```



OVERVIEW



EFFICIENCY

INSTRUCTORS CORNER

Is it possible ... make my REPORT "run faster" ?

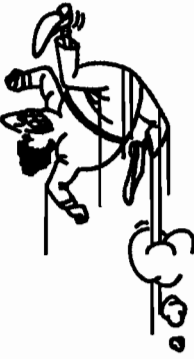
TIPS & TECHNIQUES

- . QUIZ: Tag Sort Technique
- . QTP as a front-end to QUIZ

EFFICIENCY: Other options

"Enabling your POWERHOUSE environment"

Is it possible to make my REPORT "non master"?



EFFICIENCY

My standard reply: Have your ever tried running it at 4 AM ?

Is it possible when using IMAGE Masters [M] in my reports to reduce the time it takes to place these types of records (transactions) in a specified sorting sequence ? My masters average about 55,000 records in size.

INSTRUCTORS CORNER



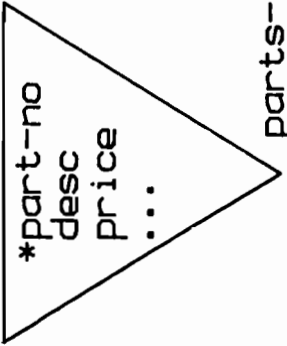
? IMAGE Masters [M]

Yes, in some situations. Making use of a KSAM file that contains the key values in a specified sorted sequence.

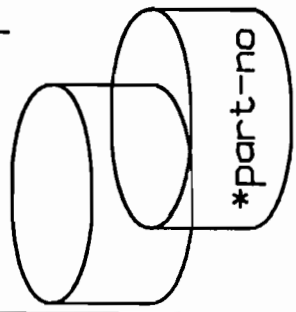


? IMAGE Masters [M]

Avoid "SORTING" Altogether !

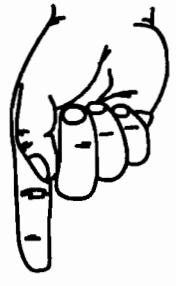


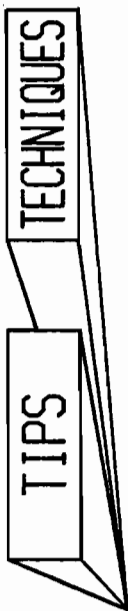
- > ACCESS parts-mstr
- > SORT ON part-no
- >
- > ; sorting of records (transactions)



- > ACCESS parts-key LINK TO parts-mstr
- > CHOOSE part-no OF parts-key
- > SORTED ON part-no OF parts-key
- >

- > ; no sorting of transactions
- > ; records are retrieved in key ascending sequence





EFFICIENCY

QUIZ: Tag Sort Technique

PURPOSE: to reduce the size of the transaction (record complex) before sorting

RESULTS: reduces processing time in sorting transactions

QTP as front-end to QUIZ

PURPOSE: to make more efficient use of machine time in producing multiple reports from the same file or collection of files

RESULTS: less CPU usage resulting in more CPU usage for other JOBS/SESSIONS; additional reports could be produced from extracted subfile in an even more timely manner

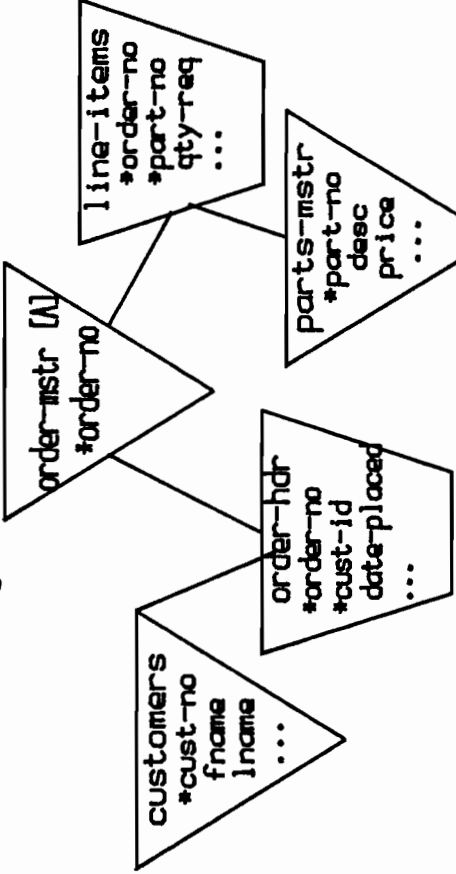
TIPS TECHNIQUES

EFFICIENCY



"Enriching your PAPERHOUSE environment"

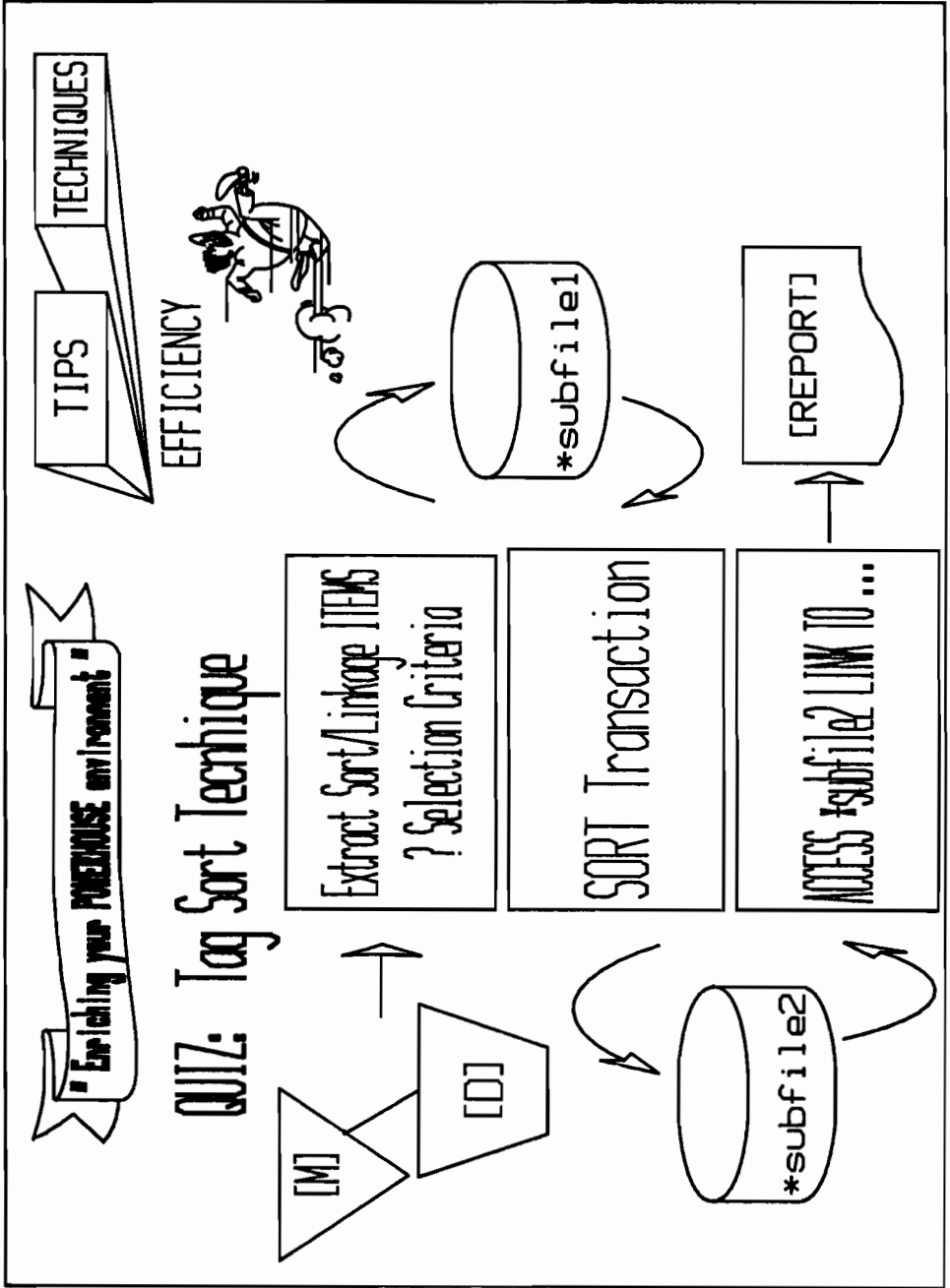
QUIZ: Tag Sort Technique
... existing REPORT ...



ACCESS customers LINK TO order-hdr
 LINK TO line-items
 LINK TO parts-mstr

SORT ON lname ON fname ON order-no
 SELECT IF state = "CA"
 REPORT ...

OPTIONAL &
 OPTIONAL &
 OPTIONAL



"Enabling your WAREHOUSE environment"

QUIZ: Tag Sort Technique
... SOLUTION ...

TIPS

TECHNIQUES

EFFICIENCY



- . reduce the size of the transaction before sorting
 - > ; pass 1
 - > ACCESS customers LINK TO order-hdr OPTIONAL
 - > SET SUBFILE NAME sub1
 - > SELECT IF state = "CA"
 - > REPORT SUMMARY cust-no lname fname order-no
 - > ; pass 2
 - > ACCESS *sub1
 - > SET SUBFILE NAME sub2
 - > SORT ON lname ON fname ON order-no
 - > REPORT SUMMARY cust-no lname fname order-no
 - > ; pass 3
 - > ACCESS *sub2 LINK TO customers &
 - > LINK TO order-header ...
 - > SORTED ON lname ON fname ON order-no
 - > ...

"Enriching your WAREHOUSE environment"

TIPS

TECHNIQUES

EFFICIENCY



**QTP as front-end to QUIZ
... SOLUTION ...**

```

: QTP
> ACCESS customers LINK TO order-header OPTIONAL &
> LINK TO ...
> SORT ON lname ON fname
> SUBFILE report1 AT lname KEEP INCLUDE lname ..... &
  IF state EQ "CA" and current-balance GT 0
> SUBFILE report2      KEEP INCLUDE lname ..... &
  IF date-placed GE 840101 AND date-placed LE 841231
> SUBFILE report3      KEEP INCLUDE lname ..... &
  IF date-placed GE 850101 AND date-placed LE 851231

> ; report1 will need to be sorted by customer number
    
```


"Enriching your POWERHOUSE environment"

TIPS

TECHNIQUES

EFFICIENCY

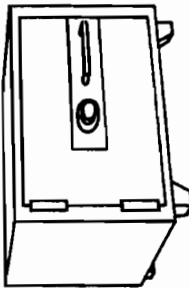


EFFICIENCY: Other Options

- . Efficient POWERHOUSE coding
 - . Linkage
 - . DEFINES efficiently
 - . SELECT filename IF ; when primary file
- . 5.01
 - . Performance enhancements
 - . Additional features
- . Q-SORT / OPT
 - . Faster sorting algorithm
- . PDQ/QUIZ / Bruce Toback/TYMLABS
 - . compiles QUIZ to machine instructions; bypass the reporter phase (interpreter)

"Enriching your POWERHOUSE environment"

OVERVIEW



SECURITY

INSTRUCTOR'S CORNER TIPS & TECHNIQUES

Is it possible .. RECORD SECURITY ?

- Password Security
- Access by Part No.
- LDC's

SECURITY: Other options



EFFICIENCY

INSTRUCTOR'S CORNER TIPS & TECHNIQUES

Is it possible .. make my REPORT "fun faster" ?

- QUIZ: Tag Seri Technique
- QTP as front-end to QUIZ

EFFICIENCY: Other options



CUSTOMIZATION

INSTRUCTOR'S CORNER TIPS & TECHNIQUES

Is it possible .. QUIZ Report Prompting ?

- Using DESIGNER Files for the End-User
- Custom Lookups for the End-User

A 4GL Comparison

Chris Brayman
Brant Computer Services Limited
Edmonton, Alberta

Contents

- 1.0 Introduction
- 2.0 The 4GL Families
 - 2.1 PowerHouse by Cognos
 - 2.2 Speedware by Infocentre
 - 2.3 Rapid by Hewlett-Packard
 - 2.4 Today by BBJ Computer Systems
- 3.0 Comparative Analysis
 - 3.1 Quick, Quiz and QTP
 - 3.2 Speedware Products
 - 3.3 Summary
- 4.0 Benchmarks
- 5.0 Conclusion

1.0 Introduction

The proper evaluation of a fourth-generation language (4GL) is a complex and time-consuming task, with the eventual selection based upon some combination of criteria that will include price, performance, support, comprehensiveness of the product, portability, ease of use and other factors.

In this paper and the subsequent presentation at the IUG in September, it is not Brant's intent to address all of these issues about all 4GLs and come out with a definitive selection of the right tools. No such presentation can or would be reasonable. For some DP shops the issue of primary concern may be price, for others ease of use, and for others portability. What may appear to be a strong suit in a product for one shop may be of absolutely no value to another.

What we do want to achieve in this paper is an evaluation of what we see as four of the most relevant 4GLs: PowerHouse, Speedware, Rapid and the new HP Today/Today product. PowerHouse and Speedware are both products Brant has done major development work with and our evaluation will be based largely on their use in large production systems. Rapid is by and large not a product we have found to be in the same league as PowerHouse and Speedware, but it is widely installed and used. HP Today is Hewlett-Packard's recently-announced 4GL for the HP9000 under UNIX, and "Today" is the sister product on the HP3000, marketed by BBJ Computer Services of Australia. The rumour mill is indicating that HP may provide "HP Today" on the HP3000 not too far down the road.

The evaluations will focus on the nature or family of products that comprise each of the 4GL tools, programmer productivity in developing systems and benchmark performance of the products running on the HP3000. The presentation of information and benchmarks will be primarily on PowerHouse and Speedware, with a more cursory evaluation of Rapid and Today.

Brant did do benchmarks in PowerHouse and Speedware in the fall, and since that time both Cognos and Infocentre have released new versions of their software. In May of this year we had the pleasure of meeting Ken Begbie and Tim Shelton of BBJ and arranged with them to receive a demo of the Today package, which we have had for about a month. Given the two new releases and the recent receipt of Today on the HP3000, we have not concluded our benchmarks to our satisfaction so will be obliged to present them at the IUG. What we can do, however, is address many of the other issues we have discovered in our evaluations.

Before commencing, we must provide a disclaimer and make it clear that in an evaluation such as this, no one is objective, regardless of how hard they try. This type of evaluation is especially dangerous since 4GL tools are like children - you can be assured there are a large number of protective parents.

Within our own organization we have programmers that love PowerHouse and others that swear by Speedware. Further, it is important for people reading this paper to understand that Brant is a Value Added Resaler of Speedware, while we are not a VAR for PowerHouse. This can result in a bias on our part. Brant does and is performing consulting work with both of these products.

2.0 The 4GL Families

2.1 PowerHouse by Cognos

PowerHouse is a 4GL family of products by Cognos of Ottawa, Canada. The software is comprised of a dictionary, a report writer (Quiz), a screen and menu generator (Quick) and a transaction processor (QTP). Along with the basic 4GL embodied in these three products, Cognos offers PowerHouse Graphics and an end-user report writer called The Expert, a financial accounting package called MultiView, and a spreadsheet program called PowerPlan.

In this section we will not bother with further descriptions of Quiz, Quick and QTP since these products will be analysed in more detail in the next section. We will give a brief overview of some of the other products from the perspective of what they do; however, Brant has not done a full evaluation of them all.

PowerHouse Graphics

PowerHouse Graphics was released last year at the IUG in Washington. Since that time we have talked to a number of clients who have evaluated the product and were impressed.

The package automatically generates vertical bar graphs with as few as three commands. It is also capable of creating line or point graphs, horizontal bar graphs and pie charts, allowing stacking, clustering and multi-summary graphs. The product has colour and hatching pattern combinations and sort capabilities.

The Expert

The Expert was released as a Cognos end-user report writer in June of

1984 and according to Cognos World, the Cognos newsletter, it is one of the world's first fourth-generation language generators. It is designed for the totally unsophisticated end-user and works by generating or writing reports in the PowerHouse development language.

Future Considerations

According to a "Preliminary Prospectus" of Cognos Incorporated on June 27, 1986, they have announced that in 1987 they will introduce a version of PowerHouse for the IBM PC/AT and compatible personal computers. This product will provide full POWERHOUSE capability and will simulate IMAGE (or other DBMS) and will support data communications between personal and mid-range computers.

2.2 Speedware by Infocentre

Speedware is a family of 4GL tools that includes Reactor, a screen generator, report writer and a batch or transaction processing capability making it equivalent to QUICK, QUIZ and QTP. There is a menu driven application generator, Designer, which front-ends the design and writing of Reactor code. In addition, Documentor is an end-user documentation generator, EasyReporter is an end-user report writer similar to The Expert, and Speedex is an integrated version of the OMNIDEX product which provides relational access to IMAGE. The MicroSpeedware product is comprised of an IMAGE/Speedex equivalent database and MicroReactor language for the IBM PC and HP150. Lastly, Speedledger is a GL/AR/AP equivalent to MultiView.

We will leave the dialogue on Reactor until later, as well as Speedex, Designer and MicroSpeedware, as we see these products as playing more directly into the development and implementation of systems. For the sake of thoroughness, we will describe the Documentor and EasyReporter.

EasyReporter

EasyReporter is an end-user report writer like The Expert and is designed to be menu and prompt driven. The user can create new reports, modify reports, etc. EasyReporter writes the Reactor code to generate the report and is fully integrated to take advantage of Speedex key word retrieval.

The Documentor

The Documentor generates customized user manuals by reading the Reactor source code that has been generated for the application. The documentation can be "regenerated" any time applications are altered

and therefore it is always up to date. Of special note is the fact that the Documentor is sensitive to system security when generating documentation by automatically tailoring the documentation to a specific user. An impressive piece of software.

Future Considerations

The next products to be released by Infocentre are a Remote Dataset Access whereby MicroSpeedware can access data and information on the HP3000 transparent to the user. This is further enhanced through a combination of a mux and small data-switch, which allows access by several micros to one port on the 3000.

Infocentre has indicated that a graphics package is due for release sometime in 1986, to be fully integrated with the Speedware family.

2.3 Rapid by Hewlett Packard

Rapid represents Hewlett-Packard's entry into the 4GL market. It is a family of products consisting of:

- Transact - the transaction processor
- Report - the report writer
- Dictionary - the data dictionary
- Inform - the user-oriented adhoc reporter

As well, Transact has built-in interfaces to VPlus, HP's block-mode screen generator.

Rapid was designed as a programmer's tool with automated interfaces to IMAGE and the MPE file system. The interfaces to VPlus allow screens to be designed independently of the transaction processing.

Transact carries many features similar to third-generation languages and so can be easily learned by in-house programmers. The underlying philosophy behind Transact allows calls to intrinsics and external procedures, and it is very flexible to use.

Report is a very lean but easy to use report writer.

Dictionary provides many advanced utilities to create and maintain databases. There are drawbacks, however. No intrinsic interface is available and dictionaries can neither be split nor combined. The dictionary itself is a very strong product which binds the Rapid

family together.

Inform provides a very good and simple means for users to obtain adhoc reports. It works off the dictionary and allows multiple dataset and database access. Security restrictions are imposed and can be stored for later use.

The Rapid family of products provides a flexible and full-featured environment for system development. The recent release of System Dictionary adds a great deal to the programming environment and is accessible by other systems. However, Rapid does not include many of the features found in other fourth-generation languages.

2.4 Today by BBJ Computer Systems

Today is a fully integrated fourth-generation computing environment which incorporates development, production, administration and documentation into a single product.

Today runs on many machines offering a UNIX operating system and on the IBM PC.

Development of an application is achieved by interacting with a Today application called the "developer". The Developer application escorts the user through the development cycle using a series of menus and formatted screens. Help screens and windows guide the user throughout the product.

Applications, once developed, are easily changed. Today also supports multiple versions of an application where special tailoring is required for specific end-users.

Today provides an "administrator" application to allow definition of users, developers and applications along with their passwords and privilege levels.

Record layouts, report headings and, optionally, screen data field specifications are dictionary based. Modifications to the data dictionary are reflected in all applications. Today provides a comprehensive reporting facility for the automatic documentation of all applications, for the developer, and of the system for the site administrator.

Report definition is flexible using formatted screens for report format, selection criteria, sorting and print options.

An application user may at any time be placed in training mode which

does not allow him to add, modify or delete information from the database.

Extensive use of formatted screens and logic tables makes Today a product which is easily learned and simple to use. Today supports its own lower level procedural language to enhance its logic capabilities. This Basic-like language contains forty commands and provides an easy interface to the full flexibility of a procedural language.

Today offers a flexible and full-featured environment for application development on either the HP3000 or the microcomputer and porting between environments requires minimal changes.

3.0 Comparative Analysis

As a data processing service organization offering a variety of custom software solutions in the HP marketplace, increasing system development productivity through the use of fourth-generation programming languages is imperative. The high cost of custom software development using third-generation languages such as Cobol are prohibitive. Internal estimates provided by our programming staff indicates a productivity gain of between five and ten times depending on the 4GL used. Hence, our rationale for a commitment to 4GL technology and the ongoing assessment of these products.

The objective of the discussion that follows is to provide a comparative analysis of the 4GLs PowerHouse, Speedware and Rapid. PowerHouse and Speedware will represent the focus of the comparison. Each of these products has been used in small and large scale application developments. Parallel evaluation of the products in the same application has also provided very specific qualitative and quantitative assessments.

The basis of our evaluation of PowerHouse and Speedware has been made simpler as a result of the development of a single large system (over 300 screens, 100 reports and substantive transaction processing on files in excess of 100,000 records) done in both PowerHouse and subsequently in Speedware. I feel it is important to point out, for those few who may not already recognize the maturity, power and flexibility of 4GLs, that their use in systems like this has demonstrated to us that their "day" has come.

More limited analysis of the Rapid products in specific areas of our applications has demonstrated clearly that it is not in the same class as PowerHouse or Speedware. Significantly more effort, measured by an increase in development time and source code lines

used, was required in parallel developments.

My evaluation of PowerHouse and Speedware is based on the results obtained from application developments in the past three years. The perceptions of various product strengths and weaknesses derive from performance in our specific application environments. As such, I do not pretend that we have evaluated every aspect of PowerHouse or Speedware, or that our results are relevant to all application environments.

Our investigations of PowerHouse 5.01 and Speedware 4.00 versions continue. The impact of these new versions on our existing applications and new developments will feel its effect in the coming year.

3.1 PowerHouse

Let me begin by outlining what I feel the high points or comparative advantages of using PowerHouse. Ease of use an important consideration in evaluating a 4GL. PowerHouse shines in this area. The programmer does not require months of experience to be productive. Our junior programmers are typically writing their first reports and simple screens in the first week. In the hands of more experienced programmer analysts, prototype systems are being produced in a matter of weeks.

PowerHouse has shown system productivity gains over Cobol of at least ten times. The number of source lines required in report and transaction processing tasks is one-fifth to one-twentieth of Cobol. On-line screens will range anywhere from one-tenth to one-twentieth the number of source lines.

Productivity gains using PowerHouse are also enhanced by excellent syntax and execution time messages received from Quiz, Quick and QTP. Source can be directed to an on-line prompt and meaningful warning and error messages guide the programmer toward the correct syntax. The programmer can design a program interactively with line by line syntax control and save the results when finished. In addition, messages often go further and assist the programmer in constructing logic that seems reasonable. Typically, if the programmer has an error-free compile, it works.

PowerHouse productivity gains go beyond the raw savings in required source code statements. The product provides a dramatic reduction in the post-debugging period after first compilation. So often with third-generation programming and, to a lesser extent, other 4GLs (including Rapid and the Reactor language of Speedware), we spend



time chasing down unobvious IMAGE or other program errors. PowerHouse programming virtually eliminates the so-called program debugging period. The price of this control is some reduction in flexibility.

PowerHouse programs communicate with an application dictionary during compile and execution phases. Centralized definitions of data files, records, elements, global options and numerous file and element attributes provide a comprehensive application dictionary. Redundant attributes such as headings, picture clauses, range and match pattern editing need to be defined only once.

Quiz and QTP allow for the creation and accessing of labeled MPE subfiles containing a mini-Qshemac definition of all data stored in the subfile. Since subfiles may include calculated temporary elements and summary operations, they provide a powerful processing feature in our more complex application tasks.

PowerHouse date elements can be stored as logical integers by using the first seven bits for the year, the next four bits for the month and the last five bits for the day. Using the QDATE storage technique, we have saved four bytes in comparison to six byte character or zoned date fields.

An audit file can be automatically produced to record changes against database files in Quick screen processing. Neither Speedware nor Rapid provides this very common application requirement as an automatic feature.

Quick screen processing allows for a wide range of file types to be defined. Through the use of primary, secondary, reference, alias and designer files, complex and flexible processing can be accomplished. Very few of our on-line application requirements could not be handled using Quick. The flexible file structure and numerous automatic and designer-written procedures allow excellent file and field element processing. Where necessary, the link to external subroutines written in Cobol, SPL, etc., is easily handled.

The use of QTP for database manipulation, including complex conversions and restructuring, is not equalled in any other language we have used. In database conversions, we typically extract to labelled subfiles and manipulate the data on the way down and/or the uploading phase. Data items can be repositioned, converted to a different storage type, calculated and much more. All of this is accomplished with remarkably few lines of code.

In report and transaction processing tasks, Quiz and QTP provide detailed statistical measures of the processing done. Both in

on-line and batch processing, Quiz provides a measure of lines printed and records selected and sorted. QTP provides statistics such as the number of records read from the various files, sorting statistics and records added, updated, modified and deleted in various files.

These statistical measures have been extremely useful in program evaluation, data maintenance and integrity, and immediate recognition of any processing abnormalities. An equivalent feature is not available directly in either Speedware or Rapid.

PowerHouse source statements are very free format and in readable English with no cumbersome delimiting characters such as periods, commas and colons.

In summary, PowerHouse offers definitive power and system development gains for the application programmer. The product lends itself well to prototyping methodologies and represents a powerful fourth-generation language. There are numerous other features in Quiz, QTP and Quick that deserve mention in this evaluation. However, I have highlighted only those product advantages that have made substantial differences in many of our PowerHouse application developments.

3.2 Speedware Products

Speedware has shown system development productivity gains over third-generation Cobol of at least five times. This assessment is based on the use of the Reactor programming language. These productivity gains are further enhanced by the use of Designer, which greatly simplifies the formatting of screens, reports or other functions and significantly reduce syntax and other errors. Although not conclusive, we feel that use of the Designer product can increase productivity of Reactor quite significantly. The current release of Designer demonstrates substantial improvement and flexibility over the previous release, but we have not yet had the opportunity to develop a very large system solely in Designer.

Reactor source files can be maintained in a single specification file and will contain both on-line and batch systems. The specification or "spec" file can have menus, screen programs, report programs, transaction processing programs, job files and more.

A special pre-compiled object code is maintained in the user labels section of the source file. This ensures compatability of production source and compiled programs, and combined with the over all spec file structure has given Reactor a substantial performance edge over

all other 4GLs for on-line screen processing.

Automatically-generated modes and various access menus bring a standardization to on-line system formats. Speedware systems tend to look and operate the same. This has proven an advantage in user training and acceptance on large on-line application systems.

Reactor on-line editing and processing by modes of processing has dramatically decreased the number of required source code statements in screen programs. A specific screen application developed originally with 860 lines of PowerHouse Quick code required only 176 lines of Reactor code. This ratio was consistent with other screen programs converted where some complex editing and processing was involved. As such, the number of reactor source code statements required in screen programs is one-tenth to one-thirtieth that of Cobol. It should also be noted that the resultant screen contained numerous features, such as extensive "Help" not previously coded in the Quick screen.

Reactor report and transaction processing programs require one-fifth to one-fifteenth the source lines of Cobol. Comparative to Quiz and QTP, Reactor requires marginally more lines of code to write the equivalent program. However, full availability of conditional logic in the report writing of Reactor can eliminate the need for multi-pass processing which is required with Quiz. On large files this can be significant.

Speedware's on-line help feature that provides automatic English narrative interpretation of field match patterns and other editing characteristics is well received by our users. Automatic display of reference file look-ups combined with the other on-line help without any programming effort increases development productivity and substantially enhances the "user friendliness" of the system.

Reactor allows application security at the menu, screen and field options levels. In addition, security is adaptable by user group and modes of processing at all three levels. Users only "see" the menu selection of fields they have access to. These flexible security features have allowed single screen processing programs in tasks with more than one Quick screen program.

Reactor has demonstrated greater flexibility in report and transaction processing tasks. Report programs that produce multi-level percents of subtotals requires several passes and complex work-around logic with Quiz. Reactor handles the task with a simple one-pass program.

Reactor report and transaction processing programs can create a variable number of extraction work files for each record read in primary and secondary linked files. As well, a variable number of print records per extraction record can be formatted for output. Both extraction and print phases can be further controlled through conditional DO WHILE logic. These features have provided impressive flexibility in more complex application tasks. The use of replacement parameter values, which can be set programatically or by user prompting in passing control information between screens and reports, increases dramatically specific complex processing capabilities over all other 4GLs we have seen.

The Designer product is a menu driven front-end to Reactor, Speedex and the Documentor that allows the user to develop systems quickly and effectively. Designer has been enhanced in the Speedware 4.00 release and allows full and flexible access to the Reactor language logic while significantly reducing development time, syntax errors etc.

Designer has the ability to operate on an existing database or design one depending on the requirements. In addition, modifications to the database during development are facilitated by this product including the transfer of test data from old to new. This can substantially reduce the time required to modify prototypes in many applications.

Speedex provides automatic maintenance of the OMNIDEX relational data retrieval system. This provides an enhanced inquiry system and a powerful selection facility for adhoc reporting systems. As an example of the speed and flexibility of this product, in an application file with over 120,000 records, where several fields had been Speedexed to allow relational and keyword access, a single value was entered in one of the keys which resulted in the selection of 127 records out of the 120,000 in less than one second. A second test selected 5,818 records in approximately one second, and then a subset of that set was selected containing ten records. The time required was again less than one second. These tests were done on a Series 42 with 35 sessions.

Speedex does have a price. The amount of disc space required to store the indexes increases incrementally with the extent of your Speedex environments. As well, batch transaction processing programs that affect the value of Speedex-related fields suffer significant performance degradation. Benchmark analysis would indicate anywhere from one-hundred to two-hundred percent more CPU and elapsed time required in these batch environments when updating fields that have been Speedexed. As with any retrieval tool, good database design and product use in the transactional data will minimize this effect.

MicroSpeedware is an application development system for the HP150 and IBM compatibles. MicroSpeedware consists of two products: the MicroReactor version of the 4GL Reactor language and the Speedbase database management system, and integrates your mainframe and personal computers by allowing portability of both applications and databases between the HP3000 and the personal computer.

3.3 Summary

The Reactor programming language is a powerful fourth-generation programming language. Speedware on-line systems are fast and responsive. System development productivity gains are not limited in very complex application environments. People performance comparisons using PowerHouse or Speedware has demonstrated significant gains over other third or fourth GLs we have used. Both products allow decreases in source code requirements and development time in similar application environments. Each product tends to have its own qualitative strengths and weaknesses, depending on the application. We have not developed any overall preferences. It depends on the individual application.

Speedware offers greater flexibility in batch reporting and transaction processing. On-line Reactor systems are responsive and faster but less flexible than PowerHouse Quick. The inclusion and maintenance of the Speedex product provides impressive relational views of IMAGE databases without adding to the development time. There is some cost to be paid in disc space and reduced batch performance.

Designer and the Documentor broaden the Speedware line in areas not yet addressed by PowerHouse. MicroSpeedware and the portability of Speedware systems from micro to mini may be a significant consideration in some environments and has already been used by Brant in the delivery of "micro solutions" for several clients.

PowerHouse has decreased the debugging aspects of programming and is therefore easier and more friendly to use than Reactor. Features such as the application dictionary, subfile processing, useful syntax, program error messages, and statistical measures of report and transaction processing make PowerHouse seem more established.

4.0 Benchmarks

Brant is currently completing benchmarks of the various products including PowerHouse 5.01 and Reactor 4.00. Report and transaction batch programs are being run against production database of between 30,000 and 100,000 records.

Benchmark testing is being performed on our HP3000 Series 42, comparing PowerHouse versions 5.01 and Speedware version 4.00. Benchmark programs were not constructed only for the purpose of these tests. Report and transaction processing programs running in a production environment using Quiz and QTP were converted to Reactor code and vice versa.

Benchmark runs are being performed with no competing sessions or jobs and all output is verified between the same report or transaction processing task in all languages before benchmarks are recorded.

For PowerHouse we are doing tests on both the privilege mode and non-privilege mode versions of Quiz and QTP. Speedware does not have a privilege mode release so tests are only with Reactor. It should be noted that privilege mode Quiz and QTP have shown gains in performance only in serial processing. Other benchmark programs that involve chained reads or less than 200 records processed in the primary file showed no difference between privilege mode and non-privilege mode versions.

We have also noticed that the new Quiz and QTP 5.01 versions have shown significant performance improvements over the previous release and are consistently outperforming the new versions of Reactor 4.00.

Although more difficult to quantify, we are providing benchmarks of on-line response of Quick versus Reactor as well as Today and Rapid. At this stage, Reactor 4.00 is consistently outperforming Quick 5.01 in this regard.

All benchmarks are being carried out on the same HP3000 Series 42 under the same conditions. Measurements of CPU and elapsed time on reports, transaction programs and on-line screen processing programs of low, medium and high complexity are underway.

5.0 Conclusion

It may appear to be a "pat" answer, but taking a look at our own situation in a production environment, we are using Speedware applications for on-line systems and Quiz and QTP for more simple but large batch applications. Not everybody can afford both or all four 4GLs, but there appears to be advantages to each in different areas.

It is Brant's intention to provide representatives of HP, BBJ, Cognos and Infocentre with copies of this paper, and more importantly, the benchmarks, for their comments and responses. We realize that this paper has been light on the Today product and on Rapid - Rapid

because it appears non-competitive and "old news"; Today because of the late date of our access to the product.

A supplement to this paper will be provided at the IUG, covering both additional qualitative and all of our quantitative information.

PERSONAL COMPUTERS SOLVE 4GL PROBLEMS

Beth C. Baerman
Charles E. Warzecha
Gateway Systems Corporation
2400 Science Parkway
Okemos, MI 48864

ABSTRACT

In an effort to achieve higher productivity and lower cost, many data processing departments have turned to fourth-generation languages. Performance, however, is a common problem. Human productivity gains can be counterbalanced by hardware degradation. Whether due to the overhead structures involved in supporting the languages' automatic facilities or their applications' interpretive code, fourth-generation languages are characteristically slow.

In general, fourth-generation languages are also inflexible. They offer a limited range of programming features and do not recognize the necessary logic constructs to create complex applications. Thus most high-transaction, production applications not only create performance bottlenecks, they are also very cumbersome to develop.

More advanced techniques yield solutions. In terms of 4GL performance and flexibility, this means utilizing the personal computer in combination with the HP 3000 for developing and executing high-volume applications. This is accomplished not by downloading parts of the database and processing offline, not by developing applications on the PC to be executed on the host, but by processing the bulk of development and execution on the PC while maintaining the database online on the HP 3000.

This paper will discuss, in detail, true PC integration -- in terms of both hardware and fourth-generation software. It will present PC integration as a solution to the common problems of fourth-generation languages.

SETTING THE STAGE

The dire situation of the data processing industry is well documented. Computerworld reports that by 1990 three computers will be delivered for every available programmer. Supergroup estimates the average application backlog to be two to five years, and reports that this figure is misleading; it does not include the "invisible backlog,"

needed applications not even requested because of the current backlog.

Furthermore, the known backlog is growing at an increasing pace. As users become more comfortable with computers and systems, they demand more. They want automated access to all the information they need to do their jobs. They also want existing systems fine-tuned, all in an effort to increase their own productivity.

For the programmer, there appears to be little hope. Programmers spend 60 to 80 percent of their time maintaining and changing existing applications -- leaving little time to complete new projects.

Fourth-generation languages have been hailed as the only practical solution to this problem. Cutting development time by significant proportions, 4GL's allow programmers to accomplish more in their limited available time.

But as more and more businesses turn to 4GL development tools, several problems have become evident. Chief among these is performance. At run time, fourth-generation languages are slow and consume significant amounts of the host processing resource.

The key to solving this performance problem is the PC. Integrating the PC into the application development and execution environments relieves the host of a true processing burden -- interpretive 4GL code. Through PC integration, each station executing a 4GL application performs most of the application's processing locally. Application screens are also stored locally; line traffic and response time are reduced. Only the database resides on the host, where it is always online and its security and integrity can be maintained.

PC'S IN THE CORPORATE ENVIRONMENT

Users who could not get timely access to the corporate mainframe and those who needed applications readily available only on the PC brought personal computers from the home environment into the working world. PC's were used to circumvent the overloaded DP department by bringing applications and processing to the individual, stand-alone level.

But to effectively do their jobs, these users needed an efficient way of accessing the data in the corporate database. Terminal emulation and file up- and downloading were responses to this need. As these techniques improved,

PC's became more commonplace in the corporate environment.

But problems remain. When users emulate terminals with their PC's they lose the benefit of having their own PC processors. They are again subject to the slowdowns of the host computer. Additionally, users downloading parts of the host database create significant problems for the data processing department in maintaining security and data integrity.

Through this evolution of PC's in the corporate environment, three essential components of true PC integration have become evident. First, to effectively integrate personal computers with a corporate minicomputer or mainframe, the PC must retain its processing capability. When the PC emulates a terminal, performance is sacrificed for the user, and a greater processing burden is placed on the host computer.

Secondly, database integrity and security must be maintained. Transfer of the corporate database must be strictly controlled or, under optimal circumstances, eliminated completely. Users should always be accessing and updating current information; multiple PC users working with multiple copies of the database makes this nearly impossible. Sensitive information must also remain under the strictest security; downloading takes the database outside the control of the DP manager.

Finally, true PC integration involves the developer as well. The programmer can also reap the benefits of a dedicated processing resource. Application development, especially processing compiles, on individual programmers' PC's can result in faster response times not only for the programming staff, but for anyone with a task on the host computer.

THE 4GL PC COMBINATION

The prospect of integrating PC's into corporate data processing in conjunction with the use of a fourth-generation language offers a very attractive alternative in dealing with the MIS backlog situation. As an added benefit, end-user productivity also increases.

End users gain greater functionality from their PC's. With corporate applications actually running on the PC (while accessing the host database), response times improve greatly. Users also gain the flexibility inherent in personal computers. When not processing corporate applications, the PC's can be used for stand-alone applications such as spread sheets and word processors.

Programmer productivity increases by benefit of both software

and hardware solutions. The 4GL contributes a much faster method of developing applications. PC integration gives the programmer his or her own processor. Development occurs in an entirely stand-alone environment. Even applications utilizing the corporate database at run time are developed entirely on the PC. Programmer access to the corporate host computer is limited to testing completed applications.

PC integration can solve, in general, the performance problem of fourth-generation languages. From the perspectives of full utilization of the PC and reduced utilization of the host, applications run faster.

PC'S AND 4GL INFLEXIBILITY

A second and related 4GL problem is that of inflexibility. A common complaint is that the constructs of fourth-generation languages do not allow for the development of complex applications. In the "real world" these are the applications in common use.

To handle complex situations, the 4GL product must be more complex -- not necessarily in terms of use, but in terms of structure and "behind the scenes" processing. In a traditional host-based 4GL environment, this means an added burden on the host processor. Thus in many cases, flexibility is sacrificed for performance.

Admittedly, the degree of flexibility in a 4GL is largely a feature of the individual product. However, fourth-generation products that achieve true PC integration have an added processing resource to combat the inflexibility problem. The PC processor can handle the added burden of increased 4GL flexibility and functionality without sacrificing performance.

With the problems and the solution identified, the issue becomes finding a fourth-generation language has everything. It must accomplish PC integration in application development as well as execution, while offering the needed functionality and flexibility.

CURRENT 4GL SOLUTIONS

Some 4GL vendors have recognized the market's movement toward PC integration. They have established a trend. Vendors of fourth-generation languages are slowly migrating toward integrating PC's with their products. Few, however, have accomplished this goal.

The two most common methods of introducing PC's with existing 4GL products have fallen short of true integration. Downloading data from the 4GL application database to the PC is one method. While this allows the flexibility to use corporate data stand-alone with PC products, it creates the security and data integrity problems involved when there are multiple copies of a database. Furthermore, this method does not attempt to integrate the PC into the 4GL's application cycle.

Another method is to run the 4GL on the PC. PC versions, generally, can be used to develop stand-alone PC applications. More significantly, some can develop applications that will later run on the host. Here, the PC is integrated for the programmer. However, the problem of 4GL performance at run time remains. The applications are processed entirely on the host using terminals or PC's emulating terminals. This method is one step closer to true PC integration. But its most important element -- integrated processing -- is missing.

The offloading of host processing to the PC is the solution to integration and the solution to 4GL performance. The only product to accomplish this goal is the SYNERGIST by Gateway Systems Corporation. When the application processing occurs on the PC, only one task remains for the host computer, processing database requests. This increases the speed of the 4GL and allows for greater flexibility.

CONCLUSION

PC's are rapidly gaining acceptance in the corporate workplace. The SYNERGIST has taken advantage of this phenomenon to improve the greatest weaknesses of fourth-generation languages -- speed and flexibility.

The SYNERGIST's PC integration means that application development and execution occur on the PC while the database is kept online on the host. Dedicated processors for each work station dramatically increase processing speed. The reduction of the host processing load improves response time. And the programmers and users alike have the added advantage of a stand-alone personal computer giving them the flexibility to use PC programs.

The Future of System Development

by Richard A. Frost

FUTURE IDEAS Inc.

P.O.Box 5406 Chicago, IL 60680

Before discussing the nature of fourth generation languages, both in general and as they exist on the HP 3000, I would like to start with a brief discussion of languages prior to the fourth generation and some other concepts which relate to the fourth generation languages. To understand more fully what a fourth generation language is and what we expect of a fourth generation language, it is helpful to look at the evolution of computer languages. The first generation of programming languages was machine language. The programmers using machine language had to be highly technical to work in tedious machine instructions using binary, octal or hexadecimal code. This machine code was very machine dependent, and entire programs may have had to be modified or rewritten for other computer systems or even new releases of the same computer system. The second generation of computer languages came about with the creation of assembly languages. With assembly languages, programmers could now use variables to reference certain memory locations rather than always having to manipulate memory directly as in machine language. This made programming easier although programmers still had to be very technical, especially in regards to the hardware of the machine that they were working on. Assemblers relieved some of the machine dependency, although there was still a high degree of dependency for programs written. This problem of machine dependency persisted until the creation of the third generation languages (COBOL, FORTRAN, etc.) which we know today. Third generation languages allowed programmers to more easily move from machine and machine. Although there may have been some differences in the COBOL implementation on one vendors machine vs. another, the language had well established standards to make it more machine independent. These third generation languages also relieved the burden of programmers needing to know machine dependent facts to as high of a degree. This allowed programmers to worry less about technical details of the hardware and concentrate more on the functionality of the programs they were working on.

The Future of System Development

by Richard A. Frost

FUTURE IDEAS Inc.

P.O.Box 5406 Chicago, IL 60680

When asking ourselves what a fourth generation language is, and what we expect of it, it is helpful to look at the trend of first, second, and third generation languages. Examining the trends of these languages, we find a tendency for:

- Higher level of machine independence for the language.
- Programmers being able to be less technical especially in regards to the hardware on which they worked.
- Languages being more human like and less machine oriented.

We would expect and desire all of these trends to continue as computer languages evolve into the future. We would also expect fourth generation languages to increase programmer productivity by allowing them to think and work in more human like languages which are less machine dependent. These languages should also exhibit a level of machine independency and should have a high level of structure to the code and data thereby creating systems which are highly structured.

While it would be nice to enforce a rigid definition of fourth generation languages, such a definition is not possible. The term fourth generation language is referred commonly to many products, some of which may or may not be fourth generation languages. Some products may not be languages at all, but merely packages which help to improve productivity. Although it is not possible to enforce a rigid definition of fourth generation languages, it is important to understand our ultimate goal and what we want out of a language. Many of the products available may not contain or fulfill all of the attributes of what we might wish out of a fourth generation language. That does not necessarily mean that they are not a fourth generation language, since many third generation languages did not live up to their full definition at times. When examining fourth generation languages, it is common to find that some of the above attributes may be lost in lieu of one which is of a

The Future of System Development

by Richard A. Frost

FUTURE IDEAS Inc.

P.O.Box 5406 Chicago, IL 60680

more primary concern. For example a fourth generation language product could forsake all for the sake of programmer productivity while perhaps being short of a fourth generation language, still be a very adequate product and meet your needs. Realize, however, that many times the choice of a fourth generation language can be a long term commitment since your applications may be tied to that language unless you wish to rewrite all of the applications. So caution should be taken not to trade short term productivity gains for other attributes which may be of value in long term strategies.

Before looking at HP3000 fourth generation languages products in detail, I would like to discuss certain terms and concepts which relate to fourth generation languages since these terms and concepts ultimately become part of the buzzwords and terminology which will be inflicted upon one when looking at fourth generation language products. First, in addition to the concept of an interpreter and a compiler, it is important to add what is most commonly referred to as a processor, which is a hybrid between the two types of execution of code. In an interpretive system the source code is interpreted each time a program is executed, as it is being executed. Hence, an interpretive system has a great deal more overhead associated with it at execution time since the code must be re-interpreted and verified as its being executed. In a compiler type of system of code execution, the source code is compiled and the ultimate outcome is object code, or machine code which can be executed directly and eliminate the need for source being interpreted at execution time hence reducing overhead. Compiler type of execution has lower overhead at execution time, although development might be slower because of the need to compile the code each time for debugging. A processor type of system of executing code is a hybrid between an interpretive and a compiled type of execution. In a processor type system the source code is compiled but the result is not true object or machine code but an pseudo-object code which is not directly executable. A processor must be then run to finish the interpretation of this

The Future of System Development

by Richard A. Frost

FUTURE IDEAS Inc.

P.O.Box 5408 Chicago, IL 60680

pseudo-object code at the time of execution. A certain degree of overhead might result as a result of this. One of the main points to consider in a processor type of system is how compiled is this pseudo-object code; and that, to a high degree will determine how much overhead is encountered at the time the pseudo-object code is executed. If minimal compilation is actually performed by a processor's compiling step, the processor system could have as much overhead as an interpreter system at execution time. If the source code in the compile phase of a processor is highly resolved, the overhead in the interpretive step of a processor system could be very minimal.

Another term to consider as part of the evolution of computer languages is the concept of code translators and generators. A code translator is simply a program or package which will translate a program from one language into another language. A code generator by some means of input generates code. This input could take the form of a data base schema, screen designs, or a pseudo-language. If this pseudo-language is highly enough developed, then a code generator might in fact be considered a translator.

Another important concept to consider when examining fourth generation language products are those of the data base management system and data dictionaries. The main intelligence or source of information about your data environment for fourth generation language products is the data dictionary. In many cases it may be helpful to examine the data dictionary and its use as part of a fourth generation language product, since to a certain degree, a fourth generation language product may not know more about your data environment than it has stored in the data dictionary. Since all access to data is controlled by the data base management system, it is also important to consider how well the data dictionary interfaces with the data base management system. Other complications which sometimes arise are difficulties from having multiple data dictionaries from multiple vendors. The management of these

The Future of System Development

by Richard A. Frost

FUTURE IDEAS Inc.

P.O.Box 5408 Chicago, IL 60680

data dictionaries being synchronized with one another is very critical. Sometimes interfaces for maintaining multiple data dictionaries simultaneously, especially the case of dictionaries from different vendors is not addressed and must be done totally manually.

One of the newest concepts and terms which has filtered into the fourth generation language arena is that of artificial intelligence. Artificial intelligence being the ability of a product to make intelligent decisions as it's part in the development process. The most common approach to this artificial intelligence implementation is that of an expert system. An expert system is a program or package which has programmed into it the detailed logic necessary to make intelligent decisions about a particular well defined area of knowledge, in this case, system development. Artificial intelligence is more and more being used as a sales point to software packages. The most important thing to consider when artificial intelligence is claimed, is the degree of intelligence that the package in fact has. One could rightly claim that any fourth generation language product has some degree of artificial intelligence. Any well developed fourth generation language product will have a certain amount of knowledge programmed into it by its creators regarding the development process. This may be a minimal amount to function as a fourth generation language product or a much higher degree of intelligence. So in fact, like a true human intelligence, the concept of IQ would be important to consider here when examining the varying degrees of intelligence that a product may have.

Since fourth generation language products are themselves evolving quickly, detailed information regarding these products will not be included as part of this paper, but will be available at the time of the presentation. A detailed discussion of some of the fourth generation language products available for the HP3000 will be given at the time of the presentation. Some of the topics this will cover as part of it's examination of fourth generation language products are the use of the data dictionaries,



The Future of System Development

by Richard A. Frost

FUTURE IDEAS Inc.

P.O.Box 5406 Chicago, IL 60680

execution mode (interpretation, compilation or processing), availability on other computer systems, and other attributes of the fourth generation language products. Products which are planned to be discussed include:

| | |
|------------|----------------------------|
| CBAS | Comprehensive Systems Inc. |
| FLEEXIBLE | SAGES American Group |
| POWERHOUSE | COGNOS Corporation |
| PROTOS | PROTOS Software Co. |
| SPEEDWARE | Infocentre Ltd. |
| TRANSACT | Hewlett-Packard Co. |

This does not mean that these are the only fourth generation language products available for the HP-3000, merely that these were the products selected for the purpose of this study. Due to the rapid evolution of fourth generation language products, not all products available on the Hewlett Packard will be discussed.

As a conclusion to this discussion on fourth generation languages it may be helpful to gaze into our crystal ball and ask ourselves what do we want beyond fourth generation languages. Along this line, I think the ultimate evolution of languages should allow the user to communicate their needs to the computer system in a human like language. The concerns of data base design and implementation will ultimately be handled more and more by the artificial intelligence of expert systems. Ultimately these future systems shall be able to be ported over to different machine vendors by the availability of more advanced data base systems. These data base management systems will allow data to be accessed as it logically exists rather than in the physical design of the data base management system. It is a belief which I have sometimes heard voiced that the ultimate evolution of computer languages will do away with the need or perhaps the entire concept of a computer programmer. Although I do not see this as the final result of advanced languages. I do see that the staff responsible for designing and implementing systems will be in some ways less technical in regards to the machine design and more technical in regards to user needs and system design. This

The Future of System Development

by Richard A. Frost

FUTURE IDEAS Inc.

P.O.Box 5406 Chicago, IL 60680

design staff may be technical users rather than programmers in and of their own right. Where does this then leave the concept of the technical programmer? Although I foresee the overall technical abilities of systems design and implementation staffs decreasing, I do still see the need for a kernel of highly technical staff since when problems arise they will be masked by more and more layers of utility software, fourth generation language products, data base management and data dictionary systems. These layers may in some cases make it very difficult to complete certain tasks or to solve certain problems. One additional concept and need which may very well be rediscovered as part of the evolution of computer languages is that of the data administrator, one central person or body which is in charge of a corporation's data. Since the concept of a central data dictionary housing and controlling the access and format for data throughout the corporation is central to high level languages and data base management systems some force must insure the integrity of this data dictionary and source of information and intelligence for the products. As a result of all of this, I see the major difference in development in the future from the present being that the definition of programmer will not be lost but it will be changed and there will be a greater rift between non-technical and technical programmers and system development staff. This in itself may bring about additional challenges to the management of development systems.

The Key to Unlocking Peak Performance

by: Casey-Davis, Kimberlee

We regret that this paper
was not received for
inclusion in these proceedings.

Table of Contents

- I. Why be concerned with a formal systems life cycle?.....1
- II. The traditional life cycle: formal and informal.....1
 - A. Tasks of the System Developer.....2
 - B. Historical Attempts to improve System Development.....3
 - C. The Victorian novel.....4
 - D. Problems with the traditional life cycle.....6
 - E. Old and New.....7
- III. Changes made possible by Fourth Generation Languages.....7
 - A. Stepwise Refinement as a Way of Life.....7
 - B. The Prototype as External Specification.....8
 - C. The End of the Signoff.....8
 - D. The End of Maintenance.....8
- IV. A proposal for a "4GL" System Life Cycle.....8
 - A. Initiation.....9
 - B. Prototype and Stepwise Refinement.....10
 - C. Implementation.....10
 - D. Operation.....11
 - E. Evaluation.....11
- V. Some practical tips for implementation.....11
 - A. Project Review Committee.....11
 - B. "Be lazy in programming, not in design".....12
 - C. How much time is enough?.....12
 - D. Releases.....13
 - E. Embroidery.....13
- VI. Conclusion: Make it easy on yourself!.....14
- Index.....15
- List of Figures
 - I. Fig. 1. The Traditional System Life Cycle4
 - II. Fig. 2. A Proposed System Life Cycle9

The System Life Cycle in the 4GL Environment

Leigh Solland
Cognos Corporation
416 Brookhollow One
2301 East Lamar Blvd.
Arlington TX 76006

I. Why be concerned with a formal systems life cycle?

The maturity of any enterprise is indicated by how many emergencies its practitioners have to deal with. Even the hospital emergency room, when well run, is mostly a routine operation: broken bones and bleeding wounds are recognized and treated using standard, proven methods. A smooth, efficient "operating" environment, indeed.

The reason the science of medicine has been able to develop as it has is that it long ago recognized that, even though ailments are manifested differently in different individuals, there are recognizable characteristics that may be used to identify a known problem and provide it with a known treatment.

As the data processing industry struggles to progress from an art to a science, certain patterns emerge. Some are new and others recall older sciences, like mathematics and management. These patterns become models, and may be used to impose some structure on an amorphous problem so that we have some sporting chance at solving it.

The system life cycle is one way of dividing a large problem into a series of smaller, manageable pieces.

II. The traditional life cycle: formal and informal

All systems, computer oriented or not, have some sort of life cycle. For a particular system, this may or may not be formally understood, but it is there whether we like it or not.

By gaining a better understanding of the cycles, flows and trends of the systems we work with, we may better predict and control what happens in them. By breaking up a large problem into smaller, more easily solved problems, we can make them much easier to solve. Computer systems, as a very visible part of commercial life and a favorite topic for academic research, make a good target for such life cycle analysis.

A. Tasks of the System Developer

What are we trying to achieve as application developers? What do our employers pay us for? What gives us satisfaction in our careers? Can these be integrated with the natural life cycle of the systems we develop to achieve a synergy?

The first task of the system developer is to pin down the end user's idea of what a given system should do. After all, the application is being built to satisfy the user's requirements, not just to provide a raison d'etre for the computer!

This is rather an artistic endeavor, consisting of drawing the visions and dreams out of the mind of the user in some more concrete form.

The second task is to formalize and enhance the raw model. Here we must call upon our own learning and experience, applying the theories of mathematical modeling, bringing in the historical lessons of business application successes and failures, and considering integration with any existing systems.

Third, we must implement the formal model. Computers are a great tool, and should be used wherever they can be effective, but remember that they are only one possible solution to the overall business problems we are trying to solve.

Our fourth task is to train people to support and use the system, and to convert the data required by the system to some useful form. This involves both a one time job to get the system started, and ongoing support for the working system.

The final task of the system developer, and one of the most interesting and challenging, is to expand the use of the system.

We must be alert for new external requirements which affect our systems, such as new governmental reports. We should be aware of new developments in our own field which could be obsoleting our existing systems, and preparing to replace those in time to maintain a competitive posture and keep expenses to a minimum. Finally, we should be on the lookout for new kinds of information which may be derived from the data already collected by our systems.

B. Historical Attempts to improve System Development

Since the first programs were written, computer science has been trying to improve the efficiency and productivity of the programmer. It has long been noted that there is a huge disparity between what has been accomplished on certain projects, or by certain people. Many learned folk have attempted to derive general lessons from these experiences and apply them to further endeavors.

Many solutions have aimed at the programmer. New, ever more abstract languages have been developed, from Fortran and Cobol in the 1950's to the Fourth Generation Languages becoming more and more popular today. Chief programmer teams attempt to use the knowledge, skill or talent of the best programmers to pull the average programmers up to a higher level. Walkthroughs also focus many minds on a single problem. Tools like HIPO charts, flow charts and structured design attempt to separate the abstract, logical design from the detailed coding in a system or a program. Structured programming simplifies and standardizes the control structures in a program, making it easier to write, debug and maintain.

The next round of solutions focused directly on the end user, on the theory that if the professional system builder was a bottleneck, he should simply be bypassed. James Martin's book on the subject, "Application Development Without Programmers", appeared at just the right time to fuel the fire. Although end user oriented query languages had been developed earlier, this activity really took off with the advent of the personal computer. Soon, every user was looking for spreadsheets, relational data bases, and Fourth Generation Languages, and the professional systems people were looking for cover!

A third set of solutions developed in parallel with the two just mentioned. Mainly oriented toward project management, and recognizing that every system has a life, the formal System Life Cycle attempts to generalize and formalize the development of an application system. The methodologies derived from these theories, such as Pride, Spectrum, and SDM/70, were typically used in larger projects, where as many as several dozen people were trying to work together to develop a system. The System Life Cycle provides a single coordination point, with the aim of ensuring that nothing is missed, all decisions are approved, and everything meets at the end of the project.

C. The Victorian novel

Typically, the System Life Cycle defines eight major phases. These are broken into modules and submodules as the project develops, but provide an overall structure for the system.

1. Initiation: establishes an overall project plan, performs a feasibility study, and decides if the project is warranted. At this time, a Project Review Committee is put into place, with management authority over the project team and the project itself.
2. Requirements Definition: generates a statement of system requirements, and establishes costs for the next phase. The scope of the project is tightened up and agreed upon by all concerned at this point.

Figure I. Fig. 1. The Traditional System Life Cycle

3. Design: sometimes known as the "External Specification" phase, this phase produces program specifications, but does not actually involve any programming. Once again, costs for the next phase are estimated.
4. Development: this is usually the largest, longest and most expensive phase of the development project. Programs are written and tested, based on the previous phase's specifications. Documentation for the programs and operating instructions for the ongoing support people are produced.
5. Implementation: data conversion, parallel runs and user training are included in this phase. Theoretically, all testing is done before now, and the programs and system flow are all totally clean (!). The products of this phase are Users' Guides and converted data (and some level of comfort that the system works).
6. Operation: when the system is finally finished, it can actually go into business. This phase produces the normal system outputs, and also "Action Requests" for changes, which may be submitted by users, operations staff or system developers. Traditionally, the software developers are out of the picture following a "handoff" to the operations people.
7. Evaluation: at some point, the Action Requests get deep enough or important enough that something has to be done to the working system. In the real world, this usually happens at 3:00 A.M. as you are trying to get the paychecks to the night shift, the orders to the warehouse and yourself back to bed, while your boss is standing there wearing a tuxedo and a scowl. Theoretically, this phase produces an evaluation report.
8. Maintenance: systems need to be maintained because of bugs, changing user requirements, and other system shortcomings. Besides keeping the system running, this phase produces modification reports and further change requests, which may eventually lead back to the Initiation phase if the system finally requires replacement.

At each of these major phases, several tasks must be

accomplished. These include:

| <u>Task</u> | <u>Responsibility</u> |
|--------------------------|--------------------------|
| 1. Stage Planning | Project Team |
| 2. Plan Approval | Project Review Committee |
| 3. Management Commitment | Project Review Committee |
| 4. Execution | Project Team |
| 5. Review | Project Team |
| 6. Approval to Proceed | Project Review Committee |

This section is entitled "The Victorian Novel" because the amount of paper generated with all these reports, studies, signoff sheets, documentation for all kinds of people, and other necessities and niceties would be enough to make any word-loving Victorian author proud. However, it is not our purpose to single-handedly keep the paper companies in business, but to design and construct better application systems.

D. Problems with the traditional life cycle

The problem with handling so much information is that you get lost in it. Communications becomes difficult on many levels, particularly because people with different backgrounds, ways of thinking and vocabularies are forced together in the project.

1. Analyst - User: dealing with users who speak everything but binary, the systems analyst does his best to translate his understanding of the user's ideas from computerese into English. Unfortunately, what the user gets is a large, wordy document which is hard to read and all but impossible to understand.
2. User - Analyst: the loop works no better the other way, so the user can't tell if his ideas were grasped and his vision is being accurately modeled in the new system. However, he is expected to sign a document approving the completed work and further direction of the project.
3. Analyst - Programmer: there are many subtle decisions to be made in the analysis, design and programming of a system. The programmer, working from a specification document, often has to guess what the user or designer wanted in a given situation.
4. Programmer - Analyst: the feedback loop back to the analyst, if it exists at all, consists of project documents. The designer may not be able to tell, until

the system is finished, if the programmers are implementing the system he was describing!

As a project gets larger, the communication problems become more and more of a problem. Even on a small project, though, there can be serious consequences from a misunderstanding or omission.

E. Old and New

Old Idea

New Idea

- | | |
|--|---|
| 1. Assume users know what they want or need. | 1. Users have a general idea, but need guidance to know what is possible, how much time and money is required, and so on. |
| 2. Assume users can understand design and signoff documents. | 2. Not even the author can understand them! |
| 3. Assume the world will hold still. | 3. Not only does the world change, but users get better ideas of what they can do with data once it is available. |

III. Changes made possible by Fourth Generation Languages

The advent of Fourth Generation Languages (4GL's) was originally marked by a notable improvement in the time it took to develop systems, and then by a large improvement in the time and effort expended on maintaining them. However, there is more than a quantitative change at work here. 4GL's have actually made it possible to do development differently.

A. Stepwise Refinement as a Way of Life

Top down testing and prototyping have long been among the tools available to system designers, but how many of us get the time to write the system twice?

With 4GL programming, the prototype does not have to be discarded. Using the defaults in the 4GL, the designer can model the overall system without becoming entangled in detail. The programmer can then complete and embellish the prototype to the level required.

A philosophy of continuous improvement of the system may be adopted, rather than one of replacement.

B. The Prototype as External Specification

The prototype can be, in the 4GL environment, more than just a proof that the project can be accomplished. It becomes the core of communication between the analyst/user design team and the programmer, passing the fundamental structure of the system on for refinement and completion.

C. The End of the Signoff

Because it is so fast and easy to generate a default system using a 4GL, the end users can be much more involved in the design. Instead of having to find a subset of English which makes sense to both parties to describe the system, we find ourselves using the system itself (menus, data screens, reports) as its own description.

The user can thereby become an integral, useful part of the review and approval process, with a short feedback loop to implement changes, catch omissions and misunderstandings, and add new ideas as the system takes shape.

D. The End of Maintenance

As mentioned earlier, maintenance results from three factors: design errors, bugs and changing requirements.

The use of 4GL programming can eliminate design errors through increasing user involvement.

It can eliminate program bugs through eliminating the most complex parts of the code, improving the understanding of the programmer, and leaving time for better testing.

This leaves only requirements changes as a reason for maintenance, and we have anticipated that through our commitment to Stepwise Refinement and continuous improvement of the system. When it becomes necessary, the whole cycle is then intentionally reinitiated for a major refinement of the system.

IV. A proposal for a "4GL" System Life Cycle

The Systems Life Cycle is a convenient way to control and manage both new systems development and existing systems. The use of 4GL technology in programming makes it possible to apply these techniques in small and medium sized installations, as well as on the large projects.

However, some things will be done a little differently.

A. Initiation

The initiation phase will undergo little change. The chief task to be accomplished here is to get the understanding of management regarding cost, time and importance of the project, and to get their blessing.

An overall project plan and feasibility study should be produced. These documents should be brief, but should adequately provide a focal point for future decisions regarding the project.

Figure II. Fig. 2. A Proposed System Life Cycle

B. Prototype and Stepwise Refinement

The Requirements Definition, Design and Development steps may all be replaced by a single, iterative step, recognizing that the actual amount of work is reduced, and that there is both a flow and a feedback loop among the user representatives, designers and programmers on the project.

The key to this stage is that all the effort expended and work performed is directly aimed at the working system, not at producing dust-gathering tomes for the shelf.

1. Scope and requirements may be addressed through a prototype menu structure. This is certainly a "top down" method of describing the system, and provides useful code at its completion.

2. General design is also described through a prototype. In this case, the data dictionary is used as the focal point for data requirements and the organizing of the data into records, files, sets, and so on.

Fundamental screens and reports may be generated, with concentration on content rather than format. Simple file maintenance screens and associated reports may be developed. Throughout this stage, the end user can see the system coming to life and provide insight and input on a timely basis.

3. The working system is derived directly from the prototype. The programmer's task is to refine, complete, document, and integrate with other systems. Because users and analysts tend to concentrate on the immediate aspects of the system, screens and reports, the programmer will likely be responsible for developing the periodic processes as well (i.e., daily, weekly, monthly, annual jobs to post, reset, archive and so on).

C. Implementation

Little change from the theoretical traditional life cycle is evident here. However, the real situation may be different. In the real world, the implementation phase, formal or informal, is often a big mess as program bugs, design weaknesses and new or misunderstood user requirements emerge from the woodwork.

In the 4GL life cycle, the bugs and design problems should

be at a minimum, so that the implementation phase can be used as it should. This involves data conversion, user training, parallel runs and the shutdown of the old system, if required.

D. Operation

A couple of changes are possible in the continued operation of the system. These stem from both the increased user involvement in the system and the ongoing support of the system developers.

First, most changes in a given system result from the user's increased information requirements. These usually take the form of requests for new reports. With the 4GL at hand, the users can take responsibility for producing many of their own reports, and can work closely with the systems people on the more complex requirements (taking advantage of the relationship, knowledge and understanding developed while working together to build the system!).

Second, the system can benefit from continued, long term involvement of the system developers. The ongoing involvement is possible because of the commitment to the system life cycle, and because of the additional time available due to increased development efficiency. In any case, the users need never feel that they have been "abandoned" at the completion of programming.

E. Evaluation

Evaluation is something which actually happens very rarely in a formal sense, and yet is important in the ongoing life of a system. It should be based both on change requests and elapsed time. Evaluating a system before it becomes an emergency is a luxury most of us only dream of, but like any other planned activity, it can likely lead to cost savings as well as cutting down on "ruffled feathers".

V. Some practical tips for implementation

The Systems Life Cycle is more than just a theory for project managers: it is a collection of ideas which can be useful to anyone involved with system development. A few practical suggestions are offered here.

A. Project Review Committee

The Project Review Committee is made up of representatives of the user, developer and operations organizations. It is formally constituted during the Initiation phase, with

specific, defined scope and authority.

Besides providing a guiding hand and a conduit to management during the development of the system, this group can prove effective later in the system's life. By keeping it alive after the completion of the initial project, "short-cycle" enhancements can be addressed and a formal Evaluation phase is more likely to take place.

B. "Be lazy in programming, not in design"

While we think of a prototype as a test, not the final result of the project, we must not succumb to "throwaway" thinking. If no value is placed on the prototype, less value will be placed on the importance of good design.

Make the prototype correct, but not perfect.

Remember the following guidelines:

1. Make it work;
2. then make it pretty;
3. then make it fast.

The order is important, because if you do get interrupted before the scheduled completion of the project, you have accomplished the maximum useful work in the time available.

C. How much time is enough?

Don't spend a long time on a given project: get it done! Both the developer and the user will know more if you get a system out and use it for a while, instead of agonizing over creating the "perfect" system.

If the time frame is too large, reduce the scope of the project. This may be accomplished with a planned phased implementation (most users would rather have some of the functions available right away than wait interminably for the ideal system). Another possible solution, for very large projects, is to assign additional project teams, each with a limited responsibility for a piece of the overall solution, once the overall system is scoped. Ideally, each module should be able to stand alone, with clearly defined interfaces to other modules.

In the absence of any evidence to the contrary, six months seems to provide a good rule of thumb. Any project

which takes longer than half a year will probably have to deal with personnel changes in both user and development groups, changing external requirements and changing internal needs.

D. Releases

Companies which produce software for sale use the technique of releases, rather than constantly fixing and modifying their systems. This technique may be used effectively by in-house developers as well. It has several beneficial effects:

1. change can be planned;
2. users may be made responsible for knowing about forthcoming changes, and for making any necessary plans and changes in their own organizations beforehand;
3. fixes can be tested thoroughly;
4. program changes can be not only tested, but even documented!;
5. the state of "constant emergency" is reduced;
6. the new release may be made an "event": bake a cake and throw a party!;
7. senior management may be made much more aware that the Information Systems organization is producing, and that they are getting tangible results for their investment.

E. Embroidery

Old Ideas

New Ideas

- | | |
|--|---|
| <p>1. Design for any future needs which may arise.</p> <p>2. Program for challenge: use all instructions available in the language because that's what they are there for.</p> <p>3. Put in the "blue sky" functions: after all, we are here to please the user and he asked for them.</p> | <p>1. Change systems as change becomes necessary.</p> <p>2. Keep it simple, get it done and get on to the next job.</p> <p>3. Ask the user "Why?"</p> |
|--|---|

VI. Conclusion: Make it easy on yourself!

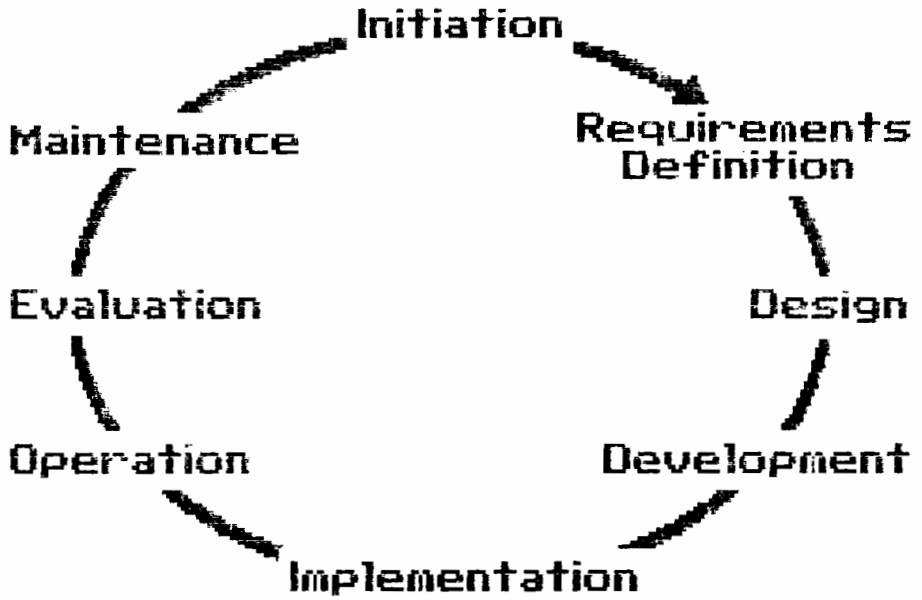
Life should not be difficult, and none of us was born to be a martyr, so make life easy on yourself!

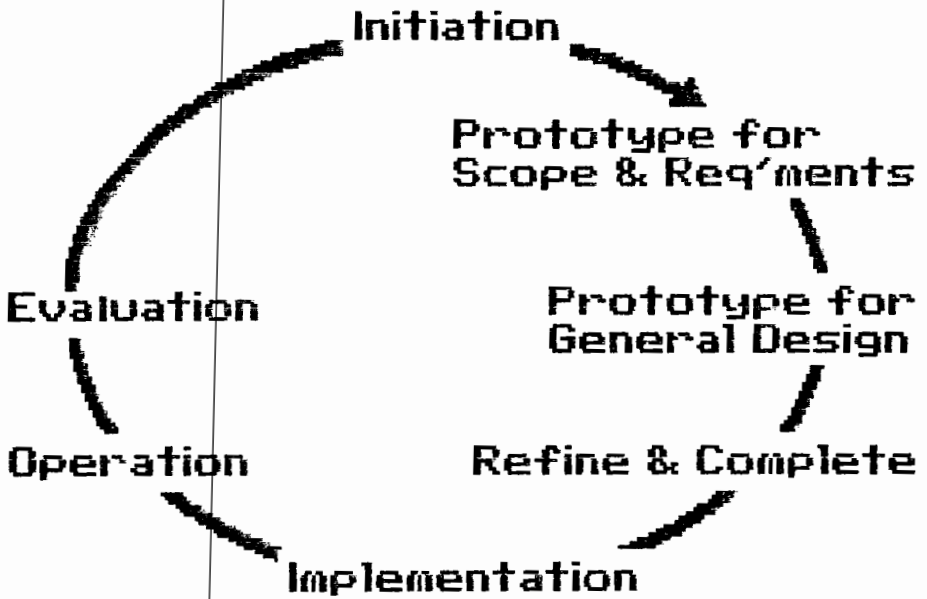
Formalize where you can. Any time you can put a pattern on a problem, you can take advantage of what those who have gone before have learned.

Determine to make your tasks routine, not emergency. Plan to get things done beforehand, and to leave the office with a light heart and no guilt when the day is done.

Delegate. There are people around you just dying to help you out and take a bigger part in your job. Encourage them.

Use Fourth Generation Language technology and System Life Cycle project organization, and enjoy the feeling of getting your systems completed accurately, on time, under budget.





Graphics In An Organization

Yvonne Temple
Hewlett-Packard
3410 Central Expressway
Santa Clara, CA U.S.A. 95051

Introduction

The field of business graphics has grown into a major business in the past few years. When DSG/3000 was first introduced, it was the only graphics package available for the HP3000, and it supported a handful of peripherals. Now the number of graphics packages and possible configurations can be confusing to the seasoned veteran, let alone a first time graphics user.

This paper is a practical guide in how to bring graphics into an organization and make effective use of the capabilities of the system. The paper will move step by step through the complex issues surrounding selection of a graphics package, i.e. personal computer versus host graphics, and how to select a package with the necessary features.

Selection of a Graphics Package

Business graphics packages fall into three general categories: charting, drawing, and mapping packages. The most important features of business graphics span all three categories, and will be discussed in detail in a later section. The differences between charting, drawing, and mapping packages are important to understand in order to determine what type of package(s) you need.

Charting packages are used to display data in a way that simplifies analysis and communicates the interrelationship of that data. The standard charting package will produce line, bar, and pie charts, as well as perform simple statistical manipulation of the data. Charts produced are generally used for two purposes: to produce rough drafts of the data for decision making, and to produce final charts for presentation of the data. Charting packages should be capable of generating both rough and final draft charts quickly and easily. Figure 1 shows a sample chart created with a charting package. The features to look for in a charting package will depend to a certain extent on the type of data you will be analyzing. Features to look for include type of charts and data manipulations supported, amount of data that can be included in a chart, and number of variables that can be manipulated. These features are fairly straightforward and will not be dealt with in detail.

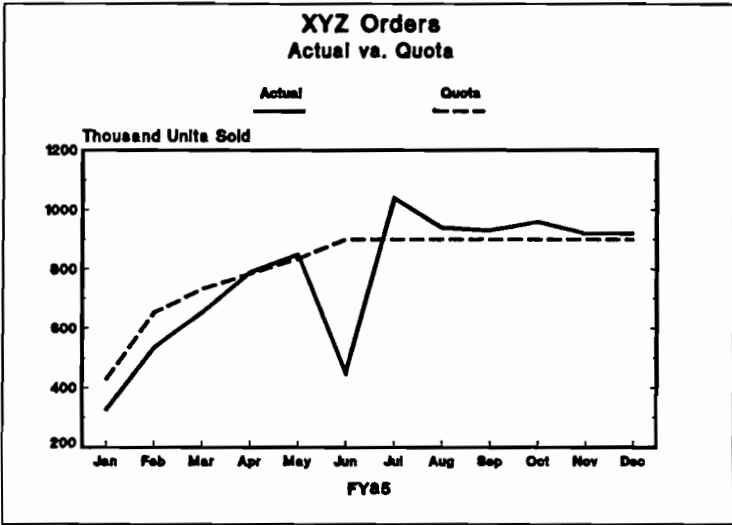


Figure 1

Mapping packages are very similar to charting packages. Mapping packages are used to display and communicate data that will be analyzed geographically. Mapping packages have just recently become available, and the advantages of geographically displayed data are not currently well understood by the average business graphics user. As maps are used more to display and analyze data trends, this area of business graphics will become more important. Figure 2 shows sample output from a mapping package.

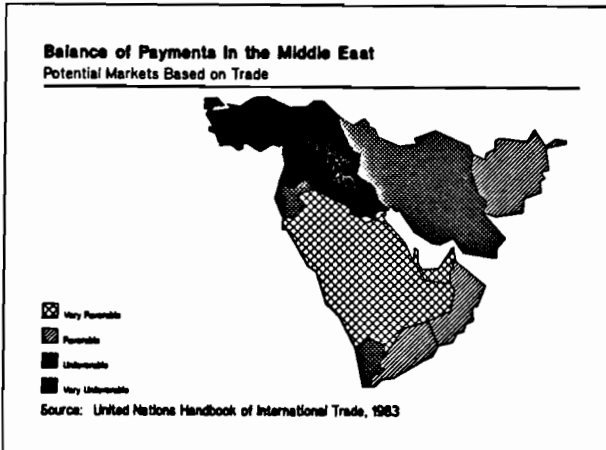


Figure 2

Drawing packages are used for a wide variety of purposes, generally centering on creating presentation graphics. Presentation graphics can take several forms depending on the formality and type of presentation: 35 mm slides, overhead transparencies, large poster-size signs, or diagrams that will later be included in a document or mailed electronically. The important thing to insure in a drawing package is that all forms of presentation graphics can be created and output easily. You may not need to create 35 mm slides when you first begin using graphics, but it is important to purchase a package that allows for increasing levels of sophistication. Features to look for in a drawing package include: ease of use, availability of templates and artist-created pictures, and high quality text capability. Output quality is one of the most important features and will be discussed in detail in a later section. Figure 3 shows a sample diagram that could be included in a presentation.



Figure 3

Different features will be important to look for depending on the type of graphics package you plan on purchasing. Clearly, data manipulation features will be important in evaluating various charting packages, but not useful in evaluating a drawing package. Two important features that span all graphics packages are frequently overlooked: output and integration.

The most important and most overlooked feature of any graphics package is output capability. The most wonderful data manipulation package in the world is worthless if you can't get accurate hardcopy of the chart created on the screen. The important things to look for in graphics output are quality and variety of supported devices. Let's talk about quality first. There are several things to take notice of when

evaluating output quality. Compare the chart in figure 1 with the chart in figure 4 as an example.

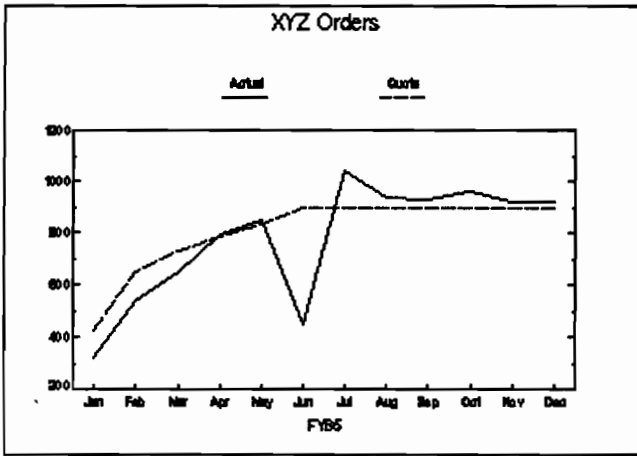


Figure 4

Both charts display the same data. The chart in figure 4 is useful for a rough draft in preparing for a presentation. The data is presented in usable form, and trend ideas begin to take shape. The chart in figure 1 is the one you'd want to show the board of directors. The chart has a polished, professional look that is important in business graphics. The key differences in output quality are: text quality, use of wide/narrow line combinations, and annotations that focus on the points to be covered by the presentation. The other key feature to look for in output that will quickly identify quality, is whether the package conforms to the idea of "what you see is what you get." This means that output looks as good on hardcopy as it did on the screen.

The second issue related to output is the variety of hardcopy devices supported. There are two things to look for here. First, does the package support a wide range of output devices. This will enable you to tailor the output device to the needs of each user. For example, it may make sense to provide each graphics user with a low resolution printer for rough draft graphics, and have a department plotter that can be used to produce the high quality presentation graphics. Attention to this detail will also enable you to grow into more sophisticated devices as the use of graphics grows in your organization.

The second issue to consider in evaluating the hardcopy device support provided by a package is quality of device support. Does the package attempt to compensate for features not supported by the device, or does the quality of output depend on the features of the device. For example, does color information get translated to grey scale when you plot your graphics to a black and white printer. Make sure that all features available on the screen are simulated as accurately as possible on each hardcopy device (another case of what you see is what you get).

The second area of evaluation for any graphics package is integration. The issue of integration can be broken into three general areas: integration with your spreadsheet (for charting packages), word processor, and other graphics packages. Integration is of particular importance when using a spreadsheet in combination with a charting package, and is a good example of what to look for in integration. There are basically three levels of integration available when working with a spreadsheet: picture, data, or worksheet integration. Picture integration would allow you to create a primitive chart using the spreadsheet, save it as a picture definition, and include that picture in other applications. This level of integration enables you to include the primitive chart "as is", but you cannot perform any further data analysis or chart enhancement. The second level of integration is data integration. This enables you to save the data from the spreadsheet and include it in a chart created by the charting package. Any special information about the data will be lost when it is brought into the charting package (for example, which data for the x-axis of the chart and which data is for the y-axis of the chart). This level of integration is more useful than picture integration because it allows you to further manipulate and enhance the chart. The best level of integration is worksheet integration. This level of integration enables you to create a rough draft of a chart using a spreadsheet and save the chart and data as a worksheet. The worksheet can then be brought directly into the charting package and all current chart attributes appear in the chart. This enables you to create a simple chart using a spreadsheet and use the graphics package to bring the chart up to presentation quality.

Integration with word processors and other graphics packages can be viewed as very similar to the example of spreadsheet integration. The details to investigate are: how completely is the information transferred from one package to another, and how much data manipulation can still be done after the data is transferred. For example, in the case of a word processor and a drawing package one level of integration would be to capture the screen image of the drawing package and include that in the word processor output. While this transfers all the information to the final document, the output quality is bounded by screen resolution. The optimal level of integration in this case would be the ability to include the drawing in the document and render the image when the document is printed. This allows the image to be enlarged, reduced or changed as necessary in successive drafts of the document.

Another area to consider for integration needs is integration across host/PC boundaries. There are scenarios that are appropriate for PC graphics packages and scenarios that are best done on a host machine. The issue of whether to use host or PC graphics is a matter of the tasks to be accomplished. A data intensive task is one where large amounts of data are generated and manipulated to produce charts on a regular basis. One example of this type of task would be tying into a corporate wide data base in order to generate monthly charts. This activity is best done on a host, using a package that can be run in batch mode (requiring minimal human intervention). A person intensive task is a task that is highly interactive. For example, producing a once-a-year chart showing the previous year's budget and extrapolating the next year's budget. Also, creating a presentation aid using a drawing package would be a highly person intensive task. These tasks are best done on a PC, where the tools available enable a higher level of interaction than a "terminal-to-host" environment. The important thing is to insure a high level of integration between the host and PC packages. For example, you should be able to include a drawing created on the PC in a document to be printed on the host machine. Tight integration of host and PC activities will become more important as graphics use grows in your organization.

HP-2680A: THE MYSTICAL PRINTER

Richard Oxford
MCI DISC
1300 Wilson Blvd. Level B-2
Rosslyn, VA. 22209

SUMMARY

This paper aims to more intimately acquaint the HP user with laser printers. This acquaintance is brought about through:

1. an explanation of technically what goes on in the box,
2. a description of the capabilities that are available to the user, and how to exploit them
3. and, an explanation of the data structures used in graphic data.

INTRODUCTION

Graphics are becoming an increasingly important part of computer information systems. Computer generated graphics range from simple bar charts, to multiple color presentation graphics, to very sophisticated solid modeling. With the advances in technology, and decreasing prices of hardware, graphic capable peripherals are becoming more affordable by the general public. They are no longer limited to large businesses with multi-million dollar DP budgets.

The decrease in price of laser printers, in particular, have made high speed printing and graphics more accessible. The need to generate graphics for the laser printer is increasing, but there seems to be only a small amount of effort being directed in this area. Here is a device capable of generating forms, logos, and signatures, as well as graphics, but yet it is quite frequently used only for high speed printing and forms generation. These printers have a lot of capability going unused, except for some packages developed by HP (TDP, HPDRAW, etc.). This "lack of use" can be attributed mostly to a "lack of knowledge" of how the laser printer actually works, and how to use it.

MCI DISC has been using many features of the 2680A laser printer for over 2 1/2 years. Many different logos, graphics, and signatures are printed during a typical job. Signatures and graphics are programmatically positioned on a page. All of this is done independent of an environment file! Most users are unaware of the ability to programmatically place graphic data, as well as text data, anywhere on the printable page. All functions performed by TDP are available to any

programmer through intrinsic calls. One only needs to be familiar with the basic operation of the laser printer, as well as some additional data formats, to be able to take full advantage of the printer's capabilities.

Since we, at MCI DISC, are currently using 2680A laser printers in our applications, most information given will be in reference to this printer, but the same basic principles apply to other HP laser printers. Since the most popular print format is 8 1/2 x 11 inches, this will be used in the examples.

Basic Laser Operation

The old saying "It's all done with mirrors" really applies to the 2680A laser printer. Although it is not necessary to know how all of the mirrors function, it is beneficial to have a basic understanding of how the laser printer works. It will then be easier to understand data structures referring to raster scan lines and dot-bit images.

The generation of a printed image by the laser printer is a rather complex procedure. The description given here is a very simplified explanation of how the image is produced, slightly modified to be more easily understood. The generation of an image in the laser printer is accomplished through a process known as electrophotography. An image is produced on paper in three basic steps:

1. Generation of the image on the drum.
2. Transfer of the image to the paper.
3. Fixing of the image on the paper.

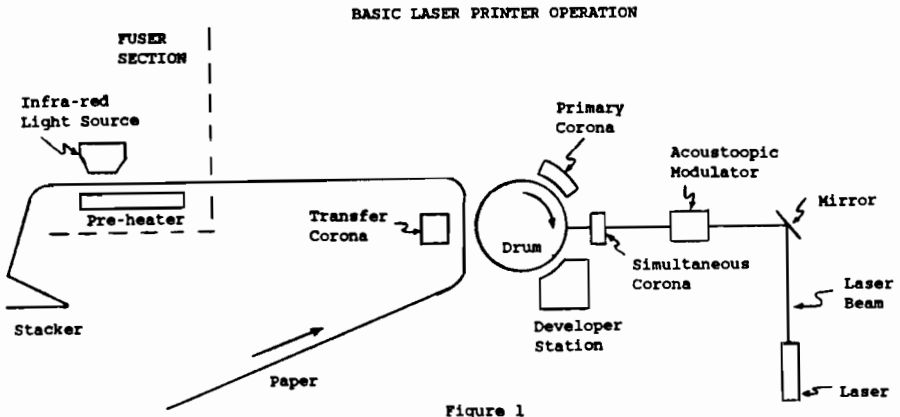


Figure 1

Figure 1 shows the basic layout of the 2680A laser printer.

actual blocking is handled by the spooler and a buffer of any reasonable size can be written.

Most users have no problem sending files to the laser, or using TDP to communicate to the laser, but are limited to the constraints of those packages. Those who have used the LPS interpreter, or have written programs using 'P'intrinsics (IFS/3000 Programmatic intrinsics), have a better understanding of laser operation. If we look one step deeper, we find that the basic method used to communicate to the laser printer is via a single callable intrinsic, FDEVICECONTROL. This is a little known, even less understood intrinsic. This intrinsic gives us the ability to change character fonts, change logical pages, position text, download pictures or graphics, and much more. Table 1 gives a summary of the control codes used with the FDEVICECONTROL intrinsic and definition of each. (All of the control code, with the exception of 139 and 144, have the parm1 and parm2 values shown in the MPE Intrinsics reference manual. The parameter definition for control codes 139 and 144 are given at the bottom of table 1.) Several of the control codes are used to set up or alter the current printing environment, load character sets and forms, or load Vertical Forms Control data. There are also codes for setting the physical page, loading logical page tables, and controlling the job. Since most of these functions are handled by the spooler when using an environment file, these functions will not be discussed here, but it is possible to have total control over the print environment even if no environment was specified for the job.

One very important concept is that of logical pages. The physical page, typically 8 1/2 x 11, can be divided into as many as 32 logical pages. A logical page is nothing more than a portion of, or a window on, the physical page. Each logical page carries its own set of specifications. They can be oriented in different directions. They can each have a different form associated with them. They can even have different character sets as their base set. Logical pages can also overlap. By simply having 4 logical pages all the same size as the physical page, but all with different orientations, you have the ability to write text on the physical page in any of the 4 possible orientations.

Now that all of the basic information has been given, let's discuss some of the function codes.

Contrary to popular belief, the laser can only have 2 character sets available at any given time. Everything else is done with slight of software. The character sets for use are selected with a control code of 128. Param1 and param2 bits 8-15, are used to define the primary and secondary character sets. Param1 holds the ID for the primary character set, while param2 holds the ID for the secondary character set. The IDs for the character sets are the font numbers as described in the environment file. The SO (%16) and SI (%17) ASCII control codes

FDEVICECONTROL control code summary

| Control Code (in decimal) | Definition |
|------------------------------|--|
| 1 | Print Data |
| 2 | File Control |
| 3 | File Open |
| 4 | File Close |
| 128 | Primary,Secondary Font Selection |
| 129 | Select/De-select Logical Pages |
| 130 | Move Pen Relative |
| 131 | Move Pen Absolute |
| 132 | Define Job Characteristics |
| 133 | Define Physical Page |
| 134 | Download/Delete Character Set |
| 135 | Download/Delete Form |
| 136 | Download Logical Page Table |
| 137 | Download Multi-copy Form Overlay Table |
| 138 | Download/Delete VFC |
| 139 | Download/Delete Picture |
| 140 | Page Control |
| 141 | Clear Portions of the Environment |
| 142 | Job Open |
| 143 | Load Default Environment |
| 144 | Print a Picture |

Controlcode = 139 Download/Delete Picture

param1 (0:1) 0 - Load a Picture
 1 - Delete a Picture

param2 (0:1) 0 - First record of a load
 1 - Continuation record of a load

(8:8) Picture identifier (0-31)

Controlcode = 144 Picture Print

param1 (0:1) 0 - Temporary Picture
 1 - Addressable Permanent Picture

(1:1) 0 - Co-ordinates X and Y are relative
 to current pen position
 1 - Co-ordinates X and Y are absolute
 pen positions on the logical page

param2 (0:1) 0 - First record of a temporary picture
 load
 1 - Continuation record of a temporary
 picture load

(8:8) Picture identifier (0-31)

Table 1

are used to switch to and from the secondary character set. (An important note: If you were using the secondary character set before execution of a control code 128, you will still be using the secondary character set after the execution, even if the secondary character set was changed.)

A control code of 129 allows the activation and deactivation of logical pages. Param2 is divided in half. The upper byte holds the logical page number to be deactivated, while the lower byte holds the logical page number to be activated. Param1 bit 0 indicates deactivate the logical page specified in the upper byte of param2, while bit 1 indicates activate the logical page specified in the lower byte of param2. Therefore it is possible to activate a logical page and deactivate a logical page at the same time with the same command. Further, all 32 logical pages can be active simultaneously. One logical page must be active at all times. Switching between active logical pages will be described later.

The command more frequently used to move between logical pages, without having to worry which ones are active, is the control code of 140. This is the page control function. It allows for a physical page eject and/or a change of logical pages. If param1 is set to a 1, a physical page eject occurs. If it is set to a 0, no page eject occurs. Bits 8-15 of param2 are used to specify the next logical page to be used. It can be the current logical page, or it can be a new logical page. Whichever logical page is specified in this parameter becomes the active logical page upon completion of the intrinsic call.

There are two pen movement commands. These are important for the positioning of data and graphics on a page. For those that are familiar with plotters, the first thing you look for is a pen up/down command. Well, there is none for the laser. The laser printer is not a plotter, and the pen command is only used to position information on the page. A control code of 130 moves the pen relative to its current position, while a control code of 131 moves the pen absolute with reference to the upper left hand corner of the logical page. The laser printer always views the logical page in its readable direction, so rotating the logical page moves its upper left hand corner (See Figure 3). The logical page is referenced in a logical page co-ordinate system with the upper left hand corner being 0,0. The X axis runs across the top of the logical page, increasing in value as you move to the right of the upper left corner, or to the right of the current pen position in a relative move. The Y axis runs down the side of the logical page. Its value increases as you move down from the upper left corner, or down from the current pen position in a relative move. The co-ordinates actually represent dots, so an X co-ordinate of 12 is actually the 13th dot along the X axis. It is the 13th dot because the co-ordinate system starts at 0. The same applies to the Y axis.

In the pen commands, param1 is the X axis co-ordinate or offset, while param2 is the Y co-ordinate or offset. These co-ordinates are expressed in 16 bit signed radix integers. The radix point falls between bit 13 and 14 of the word. Since we cannot move the pen in increments of less than a single dot, bits 14 and 15 will always be 0. To try and cut through the fog, if you wanted to move one inch, at 180 dots per inch, the parameter would be set to 180×4 or 720.

LOGICAL PAGE ORIENTATION

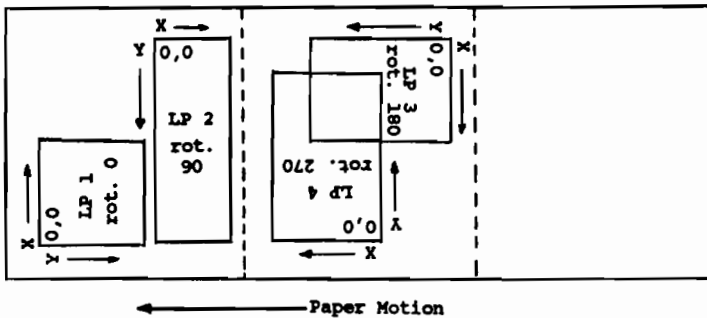


Figure 3

Now we can see how to select a logical page, select a character set, and position the pen to any point on the logical page. To print text, you simply use the `FWRITE` intrinsic. The text will be printed on the page starting at the current pen position. The current pen position is the lower left hand corner of the first character. Typically, a file is opened to the laser with carriage control. This allows the use of the control parameter in the `FWRITE` intrinsic. If mostly text positioning is being done, a code of `%320` is used. If only text is being printed, any of the carriage control codes defined with the `FWRITE` can be used to control text positioning.

When sending a page eject to the laser printer, using the `FWRITE` intrinsic with a control code of `%61`, it actually performs a logical page eject. Whether a physical page eject also occurs is dependent upon what logical page you are currently on, and how many logical pages are active. We have seen that you can activate and deactivate logical pages on command. We also know that all 32 logical pages, if defined, could be active at the same time. Logical page information is held in the laser printer in a Logical Page Table (LPT). The table has 32 entries, one for each possible logical page. When a page eject is received by the laser, it starts scanning the LPT from the current entry, looking for an active logical page. If it gets to the end of the LPT without finding one, it performs a physical page eject, and starts scanning the LPT entries from the beginning. When it finds an active LPT entry, it sets the current logical page to that entry and positions the pen to the upper left hand corner of the logical page. Of course, using the `FDEVICECONTROL` intrinsic with a control code of 140 insures that a physical page eject will occur.

Another method of printing data is to use the `FDEVICECONTROL` intrinsic with a control code of 1. This code works much the same as the standard `FWRITE`. When using this control code, `param1`, bits 8-15 are used to determine the vertical format for printing. The codes used here are

the same as %0 - %377 for a standard FWRITE with the following exceptions:

| Code | Meaning |
|------|---|
| %1 | Use the first data byte of the record for VFC |
| %61 | Conditional logical page eject |
| %62 | Physical page eject |
| %63 | Unconditional logical page eject |

Param2 bit 14 is used for Autoeject mode. A 0 indicates set Autoeject, while a 1 indicates reset Autoeject. When Autoeject is set, if carriage control causes the pen to move off the logical page, a logical page eject is automatically done and the text is printed on the next logical page. If Autoeject is not active, no text is printed and an error is written to indicate that the pen movement was off the logical page. Bit 15 is used to determine what the spacing mode will be. A 0 indicates postspacing while a 1 indicates prespacing. Prespacing causes the carriage return line feed to occur before the printing of the text, while postspacing causes it to occur after the text is printed.

The final topic to be covered is the formatting, downloading and printing of graphic data. In order to work with graphics on the 2680A, you must understand the formats of dot-bit memory words and triplets.

DOT-BIT Memory images

A dot-bit memory image is just what the name implies, a bit map of the image to be printed. Each bit of the dot-bit image represents one dot of the actual figure to be represented. Figure 4 shows an image that is 6 dots by 7 dots. In a dot-bit image, a 1 represents no dot, and a 0 represents a black dot. This means that the dot-bit image of the graphic in figure 4 would consist of 42 bits. Since data is stored in the HP-3000 in 16 bit words, this image is packed into 16 bits per word, with any remaining bits being a 1. Looking at the dot-bit map in figure 4, this can be seen. Word 0 bits 0-5 contain the dots for row 1 of the image. Bits 6-11 contain the dots for row 2 of the image. Word 0 bits 12-15 are the first 4 dots for row 3, while Word 1 bits 0-1 contain the last 2 dots for row 3. The dot image continues with Word 2 bits 4-9 being the dots for the last row of the image. Bits 10-15 of Word 2 are set to one to fill out the word. These bits will not be used by the laser printer.

This method of image representation is referred to as a "Bit Raster", or simply a Raster, image. Since the laser resolution is 180 dots per inch, an 8 1/2 by 11 image would require a bit map of $(8.5 * 180) * (11 * 180)$ or approximately 4 million dots. That would require 1/2 megabyte of memory for storage of a single graphic. Since the 2680A uses memory to store all character fonts, logical pages, and other process information, 1/2 megabyte storage for a single graphic is

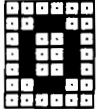
| | | DOT-BIT MAPPING | | |
|---|------|---------------------|-------------------|---------|
| | word | Dot-bit Image | | |
|  | 0 | 1 1 1 1 1 1 1 1 1 0 | 0 1 1 1 0 1 1 | %017747 |
| | 1 | 0 1 1 0 1 1 0 1 1 0 | 1 1 0 1 1 0 1 1 1 | %066667 |
| | 2 | 0 0 1 1 1 1 1 1 1 1 | 1 1 1 1 1 1 1 1 1 | %037777 |

Figure 4

unacceptable. To download 32 graphics would require 16 megabytes of memory for storage alone.

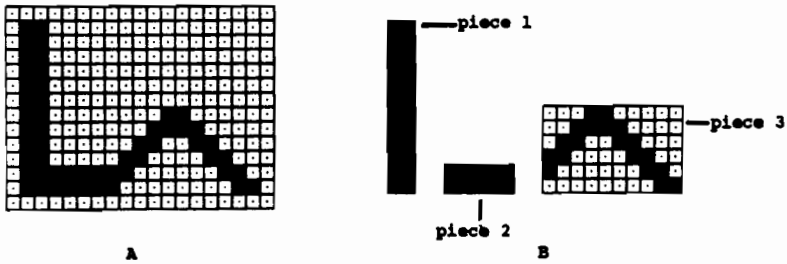
Consider an 8 1/2 by 11 page with a single line bordering all sides. If a dot-bit map were generated for this graphic, we would see that over 99% of the dot-bit map was describing white space where no image was to be printed! Now that is a real waste of memory. In order to overcome this deficiency, "Partitioned Raster" dot-bit maps were developed. In a Partitioned Raster Dot-bit map, only those dots required to define the image are used.

Partitioned Raster Dot-bit images

Figure 5A shows a graphic that is 19 dots wide by 14 dots high. In standard raster format, this would require 266 bits, or 17 words. Figure 5B shows the same image broken into 3 pieces that can be used to describe the entire image. Piece 1 requires 24 bits, piece 2 requires 10 bits, and piece 3 requires 60 bits. The "partitioned" dot-bit map in figure 5C shows word 0 bit 0 thru word 1 bit 7 hold the image for part one. Bits 8 thru 15 are padded with ones to fill out the word. Each part of the image is treated as a single image, and the dot-bit memory image must be represented in full words. Word 2 bits 0 thru 9 hold the image for part 2, and word 3 bit 0 thru word 6 bit 11 hold the image for part 3. The entire image now requires only 7 words for storage instead of 15. That is less than half that required for a standard "Raster" dot-bit image.

Now we have a much smaller dot-bit image of our graphic, but we need a way to tell the laser printer how to put the "pieces" together on the page. This is done by using what is known as "triplets". A triplet is a group of three words describing each "piece" of the graphic. Figure 6 shows the layout of a triplet. Word 0 bits 0-7 are used to indicate the number of dots there are horizontally in this piece of the graphic. This number is the 1's complement of the count. Word 0 bits 8-15 indicate the number of dots vertically in this piece of the image. We can now see that one restriction of a partitioned raster format is that each piece of the graphic cannot be any greater than 255 by 255 dots. Word 1 contains the 15 least significant bits of the memory pointer and word 2 bits 12-15 contain the 4 most significant bits of the memory pointer. The memory pointer is used to indicate

PARTITIONED DOT-BIT MAPPING



Printer Image would appear as follows

| word | Dot-Bit Image | |
|------|-------------------------------------|---------|
| 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | %000000 |
| 1 | 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 | %000377 |
| 2 | 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 | %000077 |
| 3 | 1 1 1 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 | %163760 |
| 4 | 1 1 1 1 1 1 0 0 1 1 0 0 1 1 1 1 0 0 | %174634 |
| 5 | 1 1 1 1 0 0 1 1 0 1 1 1 1 1 1 1 0 0 | %171576 |
| 6 | 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 | %077717 |

C

Triplet Words

| | piece 1 | piece 2 | piece 3 |
|---|---------|---------|---------|
| 0 | %176414 | %175002 | %172406 |
| 1 | %000000 | %000002 | %000003 |
| 2 | %000020 | %000060 | %000200 |

D

Figure 5

TRIPLET WORD LAYOUT

| | MSB | LSB |
|------|---------------------------------------|--------------------|
| word | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 | |
| 0 | - Hor Dot Cnt. | Vertical dot count |
| 1 | LSB Dot-bit memory pointer | |
| 2 | X Coordinate of left edge | MSB |

Figure 6

which word, starting from 0, in the partitioned dot-bit map that the image for this piece of the graphic begins. This pointer is 20 bits

long, because dot-bit maps for large graphics can exceed 65,000 words. This 20 bit pointer allows us to have a graphic that requires over a million words to represent. Word 1 bits 0-11 are used to indicate the position along the X axis of the entire graphic, starting from 0, that this piece of this image is to be placed.

Figure 5C shows the 3 triplets that would be used to describe the image shown in Figure 5B. Piece 1 has a horizontal count of -2 (one's complement), a vertical count of 10, the first bit describing this part of the image starts at word 0 in the bit map, and this piece of the image starts at dot 1 on the x axis of the graphic. Looking at the triplet for piece 3 of the image, its horizontal count is -10, the vertical count is 6, it starts at word 3 in the bit map, and its X coordinate is 8.

DOWNLOADING OF GRAPHIC DATA

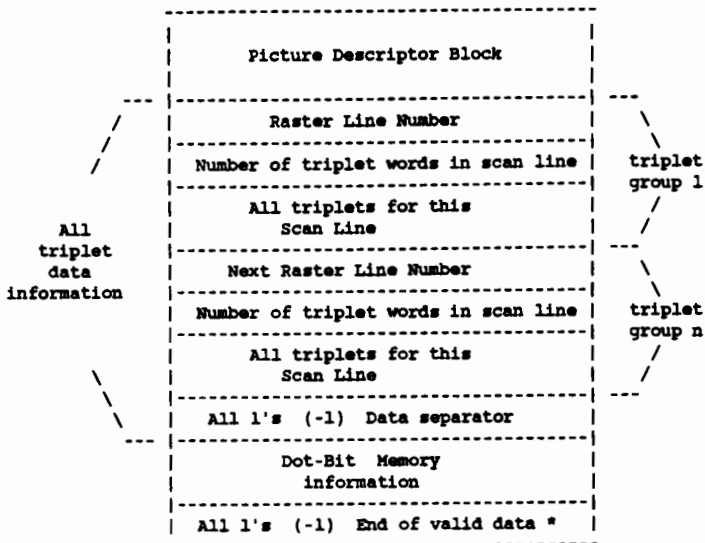
Now that the pieces to the graphic are formatted, they need to be downloaded to the laser printer. An FDEVICECONTROL with a control code of 139 is used. This is the download/delete picture function. Param1 bit 0 is set to a 0 to delete a picture, or set to a 1 to download a picture. Param2 bits 8-15 are the picture identifier. This can be an integer number between 0 and 31, since the laser can hold 32 pictures. When downloading a picture, if a picture currently exists with the same id, it is deleted, and replaced with the one being loaded. Since picture data can span more than one record, it is necessary to be able to indicate continuation records for download. Param2 bit 0 is used to indicate if the current record is the first record of this picture, or if it is a continuation record for a picture in the process of being downloaded. Bit 0 is set to a 0 for the first record and set to a 1 for all subsequent records. All records for a picture must be downloaded without interruption. There cannot be any other commands sent to the laser printer until all of the picture has been downloaded.

Figure 7 shows the layout of picture download data. The data is divided into 3 functional parts:

1. Picture Descriptor Block
2. Triplet data information
3. Dot-bit map data

The first part encountered is the picture descriptor block. This data block is used to describe the characteristics of the picture being loaded. This will be covered in detail later. The next part contains all of the triplets, in groups, that are needed to describe all pieces of the picture. Triplets are grouped together according to their starting location in the picture. The first word in a triplet group represents the raster scan line where the following triplets will begin. Recall, that when the picture was broken into pieces, each piece had a triplet describing it. The triplet described the size of the picture piece, horizontally and vertically, where the data for

PICTURE LOAD RECORD LAYOUT



*added by the spooler

Figure 7

that piece could be found in the dot-bit map, and the X co-ordinate of the left edge of that piece in the entire picture. The only thing missing that would be needed for proper placement of the piece in the entire picture was the Y co-ordinate. The Y co-ordinate is referred to by raster scan line number, so the raster scan line number for a particular triplet would be the Y co-ordinate, starting from 0, of the top line of this piece in the picture. The next word in the triplet group indicates how many triplet words there are for this scan line. If there are 2 triplets for picture pieces starting on this scan line, then this number will be 6 (3 words per triplet, 2 triplets). Following this are all of the triplets that start on this scan line. Due to limitations of the laser printer, the maximum number of triplets, or pieces of the picture, that can start on a single scan line is 255.

Following this triplet group would be another triplet group for the next raster scan line. The triplet groups are loaded in order by ascending Y co-ordinate (raster scan line number), starting from the top of the picture. Since the Y co-ordinate starts at 0 at the top of the picture, and goes thru some Y co-ordinate N, the triplet group for raster scan line 0 will be the first group and the group for raster scan line N will be the last group. If there are no pieces of the picture that start on a given raster scan line, then there is no

triplet group required for that scan line. After the triplet data information there is a -1 (all ones) data separator. This is used to signify the end of the picture description, and the beginning of the dot-bit memory map for the picture. Following this data separator, is the dot-bit memory map for the entire picture.

PICTURE DESCRIPTOR BLOCK LAYOUT

| | MSB | LSB | | | | | | | | | | | | | |
|------|---------------------------------------|-----|--|--|--|--|--|--|--|--|--|-----|--|--|--|
| word | 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 | | | | | | | | | | | | | | |
| 0 | not used | | | | | | | | | | | MSB | | | |
| 1 | Number of Triplet words in Picture | | | | | | | | | | | | | | |
| 2 | not used | | | | | | | | | | | MSB | | | |
| 3 | Number of Dot-bit memory words | | | | | | | | | | | | | | |
| 4 | Offset to left edge of Picture | | | | | | | | | | | | | | |
| 5 | Offset to right edge of Picture | | | | | | | | | | | | | | |
| 6 | Offset to top edge of Picture | | | | | | | | | | | | | | |
| 7 | Offset to bottom edge of Picture | | | | | | | | | | | | | | |

Figure 8

The picture descriptor block is 8 words long and is used to describe overall information about the picture being downloaded. Figure 8 shows the layout of the picture descriptor block. Word 0 bit 12 thru word 1 bit 15 is a 20 bit integer which indicate the total number of triplet data information words in the download record. This includes all triplets, all scan line numbers, and also the data separator as shown in Figure 7. Word 2 bit 12 thru word 3 bit 15 is a 20 bit integer indicating the number of dot-bit memory map word used for this entire picture. Words 4 thru 7 are used to describe the location of the picture origin. The picture origin is used to determine where to place the picture on a page. When a picture is printed on the laser printer, it is positioned with the origin at the current pen position. In most cases, the origin is placed in the upper left hand corner of the picture. This allows the corner of the picture to fall at the current pen position.

Figure 9 shows the data buffer that would be used to download the graphic shown in figure 5A. This image would be downloaded using the FDEVICECONTROL command with a control code of 139, param1 set to a 0, and param2 set to a 2. Buffer length for the download is 29 words. At the completion of the command, this picture would be stored as picture 2 in the laser printer.

PICTURE DOWNLOAD RECORD for
 PICTURE in FIGURE 5A
 (Permanent Addressable)

| | word | buffer data | |
|--------------------------------|------|-------------|------------------|
| | 0 | %000000 | # triplets |
| | 1 | %000016 | in description |
| | 2 | %000000 | # dot-bit memory |
| | 3 | %000007 | map words |
| Picture descriptor block | 4 | %000000 | offset to left |
| | 5 | %000016 | offset to right |
| | 6 | %000000 | offset to top |
| | 7 | %000023 | offset to bottom |
| | 8 | %000001 | scan line 1 |
| | 9 | %000006 | 6 triplet words |
| | 10 | %176414 | - |
| triplet group 1 | 11 | %000000 | triplet 1 |
| | 12 | %000020 | - |
| | 13 | %175002 | - |
| | 14 | %000002 | triplet 2 |
| | 15 | %000040 | - |
| | 16 | %000006 | scan line 6 |
| | 17 | %000003 | 3 triplet words |
| triplet group 2 | 18 | %172406 | - |
| | 19 | %000003 | triplet word 3 |
| | 20 | %000200 | - |
| | 21 | %177777 | data separator |
| | 22 | %000000 | |
| | 23 | %000377 | |
| | 24 | %000077 | dot-bit memory |
| | 25 | %163760 | map |
| | 26 | %174634 | |
| | 27 | %171576 | |
| | 28 | %077717 | |

Figure 9

To print the picture, an FDEVICECONTROL command with a control code of 144 is used. Param1 bit 0 is used set to a 1 to indicate an addressable picture, or set to a 0 to indicate a temporary picture. An addressable picture is one which is downloaded with a control code of 139. A temporary picture is one which is downloaded with this command, is printed, and then deleted from memory. Temporary pictures will be covered later. Param1 bit 1 is set to a 0 to indicate that the picture is to be printed with its origin at the current pen position. It is set to a 1 to indicate that the origin is relative to position 0,0

on the logical page. Param2 bit 0 is a 0 for permanent pictures, and bits 8-15 are used to identify the picture. If we wanted to print the picture just loaded at the current pen position, then parameters 1 and 2 would be %100000 and %000002 respectively.

TEMPORARY PICTURES

When a picture is to be printed only once, it is better to download the image as a temporary picture. This prevents the memory in the laser printer from being used to hold images that are printed only once. It also keeps the laser printer memory free to hold 31 permanent images. (One of the 32 image ids must remain free for the download of the temporary picture, but the picture is deleted after printing.) This means, that if many images are to be used only once, there is no limit to the number of pictures that can be printed during a logical job. A temporary picture is actually downloaded with the print picture command, control code 144. When downloading a temporary picture, the buffer is formatted the same as with a permanent picture except the picture descriptor block is preceded by 2 extra words, as shown in Figure 10. The first word is an X co-ordinate, and the second word is a Y co-ordinate for placement of the temporary picture. The co-ordinates are expressed in radixed integer format, the same as those used for pen movement. When downloading a temporary picture with a control code of 144, param1 bit 1 is used to indicate whether the X and Y co-ordinates preceding the picture are absolute pen positions on the logical page or relative to the current pen position. A 0 indicates they are relative, while a 1 indicates that they are absolute pen positions. As with the download picture command, control code 139, download information can span more than one record. Param2 bit 0 is used to indicate the continuation of a picture download, just as it was with the control code of 139.

EXAMPLE

Figure 11 shows a sample program, written in pseudocode, to download the picture described in figure 5. It is downloaded using a picture ID of 2, the pen is positioned and the picture is then printed. Next, a temporary copy of the same picture is printed. Notice that a picture ID is used to download the temporary picture. If a picture was currently using that picture ID, it is first deleted, and then the temporary picture is downloaded. Once the picture is downloaded, it is printed and deleted. It is important to note that the temporary picture is not truly deleted until a physical page eject occurs. Therefore, if more than one temporary picture is to be downloaded on a physical page, they must all have different picture IDs. Once a physical page eject occurs, the picture IDs of all the temporary pictures may be reused. Figure 12 shows the resulting output from the program. For clarity, the size of the image was increased. The text at the bottom of the second picture was printed using the FDEVICECONTROL 131 and FWRITE intrinsics.

PICTURE DOWNLOAD RECORD for
 PICTURE in FIGURE 5A
 (Temporary)

| | word | buffer data | |
|--------------------------------|------|-------------|------------------------|
| | 0 | %000100 | X coordinate (radixed) |
| | 1 | %000100 | Y coordinate (radixed) |
| Picture descriptor block | 2 | %000000 | # triplets |
| | 3 | %000016 | in description |
| | 4 | %000000 | # dot-bit memory |
| | 5 | %000007 | map words |
| | 6 | %000000 | offset to left |
| | 7 | %000016 | offset to right |
| | 8 | %000000 | offset to top |
| | 9 | %000023 | offset to bottom |
| triplet group 1 | 10 | %000001 | scan line 1 |
| | 11 | %000006 | 6 triplet words |
| | 12 | %176414 | - |
| | 13 | %000000 | triplet 1 |
| | 14 | %000020 | - |
| | 15 | %175002 | - |
| | 16 | %000002 | triplet 2 |
| | 17 | %000040 | - |
| triplet group 2 | 18 | %000006 | scan line 6 |
| | 19 | %000003 | 3 triplet words |
| | 20 | %172406 | - |
| | 21 | %000003 | triplet word 3 |
| | 22 | %000200 | - |
| | 23 | %177777 | data separator |
| | 24 | %000000 | |
| | 25 | %000377 | |
| | 26 | %000077 | dot-bit memory |
| | 27 | %163760 | map |
| | 28 | %174634 | |
| | 29 | %171576 | |
| | 30 | %077717 | |

Figure 10

EXAMPLE of PICTURE
DOWNLOAD and PRINT

The following is a pseudocode example of how to download and print the image shown in Figure 5A. All buffer offsets start at 1. This example shows both a temporary and permanent addressable picture load and print.

```

Bufferf = Logical 31 word buffer initially loaded with the
          temporary picture data in Figure 10

Bufferp = Logical 29 word buffer equated to Bufferf(3), which
          would make it the same as the buffer in Figure 9

Textbuffer = a logical buffer used for text data

Laser = "Laser "

***** Open the spoolfile to the laser as ASCII,NEW,CCTL,WRITE ONLY
Laserout = FOPEN(Laser,%04,%1)

***** Download the permanent picture with ID 2
FDEVICECONTROL(Laserout,Bufferp,29,139,%0,%2)

***** If there was a continuation record, the buffer and count would be
***** changed and param2 would be set to %100002

***** Move the pen to an absolute location
X = %002640      **** 2 inches (180 dots per inch) radixed
Y = %001320      **** 1 inch (180 dots per inch) radixed
FDEVICECONTROL(Laserout,Notused,Notused,131,X,Y)

***** Print the permanent picture at the current pen position

FDEVICECONTROL(Laserout,Notused,Notused,144,%100000,%2)
***** The picture is now printed with the origin at X = 2 inches
***** and Y = 1 inch.

***** Now print the picture as a temporary picture using the
***** X and Y in the download file to position the origin at
***** absolute X = 4 inches and Y = 4 inches
Bufferf(1)=%005500
Bufferf(2)=%005500
FDEVICECONTROL(Laserout,Bufferf,31,144,%040000,%3)
***** The laser will temporarily use Picture ID 3 to download
***** and print the picture. Once the picture is printed, it
***** is deleted.

***** Move the pen under the picture and print text

Move "This is the figure title" to Textbuffer
FDEVICECONTROL(Laserout,Notused,Notused,131,%5500,%5764)

***** The pen is now 1 inch under the origin of the tempory picture

FWRITE(Laserout,textbuffer,-24,%320)

FCLOSE(Laserout,1,0)

END of pseudocode

```

Example Program Results

Please note that the picture size has been increased for clarity

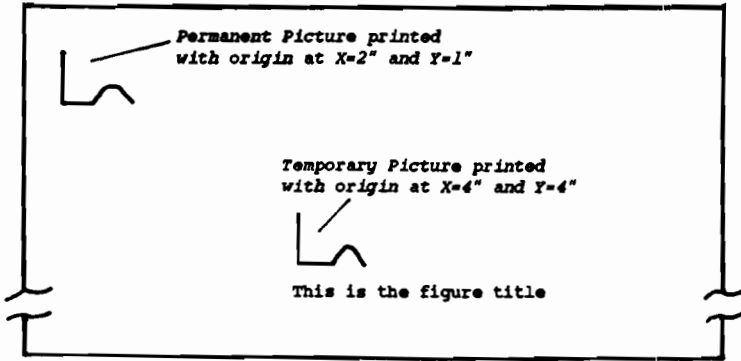
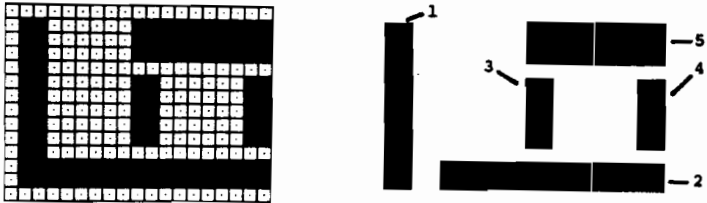


Figure 12

CONSERVING DOT-BIT MAP SPACE



| word | Dot-Bit Image | |
|------|---------------------------------|---------|
| 0 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | %000000 |
| 1 | 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 | %000000 |

Triplet Words

| | 1 | 2 | 3 | 4 | 5 |
|---|---------|---------|---------|---------|---------|
| 0 | %176414 | %175002 | %176405 | %176405 | %172403 |
| 1 | %000000 | %000000 | %000000 | %000000 | %000000 |
| 2 | %000020 | %000060 | %000220 | %000420 | %000220 |

Figure 13

CONDENSING DOT-BIT-MAPS

Since triplets are used to point to a location in the dot-bit map where a portion of a picture begins, if there are several parts of the picture with identical data, the same locations in the dot-bit map can be used. A very good example is shown in Figure 13. All pieces of the graphic can be described as an all black image. If we describe the largest black area in the image, in this case piece 2, we see that it takes 32 bits to describe. All of the other pieces of the image are smaller, and can be described using the same dot-bit map area. All of the triplets in figure 13 point to word 0 for the dot-bit map data. The only difference is the horizontal and vertical counts, as well as the X coordinate.

BIOGRAPHY

Richard Oxford has been involved with computer hardware and software for the past eleven years. His experience includes teaching computer hardware, developing software, and maintenance of hardware and software on various computer systems. He has worked with HP computers for the past seven years, and with the 2680A laser for four years. Presently he is with MCI Digital Information Services as system manager for their network of HP-3000s and laser printers.