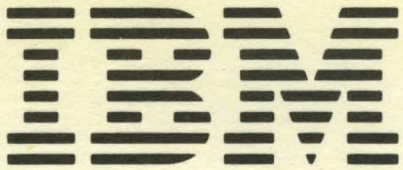
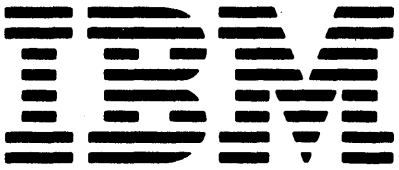


SC21-7742-3
File No. S34-34

IBM System/34
Concepts and Design Guide

Program Numbers 5726-SS1
5726-AS1
5726-FO1
5726-RG1
5726-UT1
5726-CB1
5726-BA1



SC21-7742-3
File No. S34-34

IBM System/34
Concepts and Design Guide

Program Numbers 5726-SS1
5726-AS1
5726-FO1
5726-RG1
5726-UT1
5726-CB1
5726-BA1

Fourth Edition (January 1982)

This is a major revision of, and obsoletes, SC21-7742-2. New material includes System/34 storage concepts, IFILE programming considerations, print spooling, the X.21 interface, and data processing security. Changes or additions to the text are indicated by a vertical line to the left of the change or addition.

This edition applies to release 08, modification 0 of IBM System/34 and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters.

Changes are periodically made to the information herein; changes will be reported in technical newsletters or in new editions of this publication.

Use this publication only for the purposes stated in the *Preface*.

This publication contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country. (For example, ideographic support is available only in Far East countries.)

Publications are not stocked at the address below. Requests for copies of IBM publications and for technical information about the system should be made to your IBM representative or to the branch office serving your locality.

This publication could contain technical inaccuracies or typographical errors. Use the Reader's Comment Form at the back of this publication to make comments about this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901. IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Preface

This manual is primarily for System/34 designers, analysts, and programmers who are changing a centralized operating environment to a work station operating environment. The reader should understand how System/34 functions can be used in a centralized environment. The System/34 Implementation course, S2020 provides one way to obtain this information.

The manual is divided into the following segments:

Chapter 1. Introduction: Explains the need for careful design and programming on System/34 in a work station environment.

Chapter 2. System/34 Concepts: Reinforces many concepts that are important in either a centralized or work station environment and presents additional concepts that should be understood before design and programming in a work station environment begins.

Chapter 3. Design Considerations: Describes designing displays, input documents, printer forms, files, programs, and security and integrity for a work station system.

Chapter 4. Coding Techniques: Presents coding examples that illustrate concepts explained in Chapters 2 and 3.

Chapter 5. Sample Applications: Presents a sample order entry application and sample inquiry applications.

Appendix A. Display Station Operations Requested by Basic Assembler Programs: Describes the work station data management operations that can be requested from an assembler program.

Note: This manual follows the convention that *he* means *he* or *she*.

Prerequisite Publications

IBM System/34 Introduction, GC21-5153

IBM System/34 Planning Guide, GC21-5154

Related Publications

- *IBM System/34 System Support Reference Manual*, SC21-5155
- *IBM System/34 Operator's Guide*, SC21-5158
- *IBM System/34 Command and OCL Statements Reference Summary*, GX21-7690
- *IBM System/34 Data File Utility Reference Manual*, SC21-7656
- *IBM System/34 Source Entry Utility Reference Manual*, SC21-7657
- *IBM System/34 Sort Reference Manual*, SC21-7658
- *IBM System/34 Work Station Utility Reference Manual*, SC21-7663
- *IBM System/34 RPG II Reference Manual*, SC21-7667
- *IBM System/34 Installation and Modification Reference Manual*, SC21-7689
- *IBM System/34 Data Communications Reference Manual*, SC21-7703
- *IBM System/34 Interactive Communications Feature Reference Manual*, SC21-7751
- *IBM System/34 Basic Assembler and Macro Processor Reference Manual*, SC21-7705
- *IBM System/34 Screen Design Aid Programmer's Guide and Reference Manual*, SC21-7716
- *IBM System/34 Overlay Linkage Editor Reference Manual*, SC21-7707
- *IBM 5211 Printer Models 1 and 2 Component Description and Operator's Guide*, GA24-3658
- *IBM 3262 Models A1 and B1 Component Description and Operator's Guide*, GA33-1530
- *IBM 5224 Printer Models 1 and 2 Setup Procedures*

- *IBM 5224 Models 1 and 2 Operator's Guide*, GA34-0092
- *IBM 5256 Printer Operator's Guide*, GA21-9260
- *IBM 5225 Printer Operator's Guide*, GA34-0089
- *IBM System/34 FORTRAN IV Reference Manual*, SC21-7706
- *IBM System/34 COBOL Reference Manual*, SC21-7741
- *IBM System/34 Installation Manual—Physical Planning*, GA21-9242
- *IBM System/34 Physical Planning Template*, GX21-9280
- *IBM System/34 BASIC Reference Manual*, SC21-7835
- *IBM System/34 System Measurement Facility Reference Manual*, SC21-7828

Contents

Chapter 1. Introduction

IBM System/34 is a general-purpose system that can be designed to operate in a work station environment. In this environment, the system is available to users via display stations and printers that are in their departments. A major advantage of this environment is that users have access to current, correct information in the system. Also, users can enter data directly into the system and find and correct errors that might otherwise be overlooked.

Designing a system and applications that fit your business needs requires activities such as planning displays, menus, input documents, output forms, files, programs, and security and integrity procedures. Some of this planning would be required for any type of system, but some additional planning must be done especially for a work station environment. This manual provides important information that you should know in order to design a system in that environment.

A well-designed system should have the following characteristics:

Easy to Use: System/34 operators should not need to understand how the system or its programs work. If the displays are understandable, if applications are divided into logical steps, if written operator instructions are clear, and if operator data entry is minimized, operators can be productive and make few mistakes.

Provide Adequate Throughput: Throughput is the amount of work done by the system during a period of time. You could determine for example, the number of invoices, orders, and inquiries that should be processed over a period of time and design a system that can realistically meet those requirements.

Provide Satisfactory Response Times: Response time requirements can vary significantly. For example, a 15-second response time might be adequate for an inquiry program that is used occasionally, but a 2-second response time might be required for an order entry application that is used for an hour or more at a time.

Able to Change and Grow: A well-designed system allows for future expansion. For example, the design should allow for additions of display screens, printers, and new applications. Be aware that a design might be adequate initially but might require major rework when the work load increases or additional applications are installed.

Provide Security and Integrity: System security and integrity should be planned for the system so that information required for an audit can be maintained, recovery from system failures can occur, and the system cannot be used without proper authorization.

This manual has been written to help you design a system that has these characteristics. Chapter 2, *System/34 Concepts* provides information that should help you design and code applications that use system resources efficiently. Chapter 3, *Design Considerations* provides considerations for many of the design activities that you will do. Chapter 4, *Coding Techniques* describes techniques that should be of interest to programmers. Chapter 5, *Sample Application* describes the design and development of a sample order entry application. These chapters can give you a better understanding of your System/34 and of designing applications to meet the objectives you set.

If you are an experienced designer of work station systems, you would already know much of the information in Chapter 3 and might want to skip that chapter. If you thoroughly understand how System/34 works and are interested mainly in designing the system, you might skip Chapter 2 and begin reading Chapter 3.

Binary synchronous communications (BSC) between the System/34 and some office product devices are available with an RPQ. These office product devices are:

- 6640 Document Printer
- OS/6 Information Processor
- Magnetic Card II–Communicating
- 6670 Information Distributor
- 6240 Magnetic Card Typewriter–Communicating

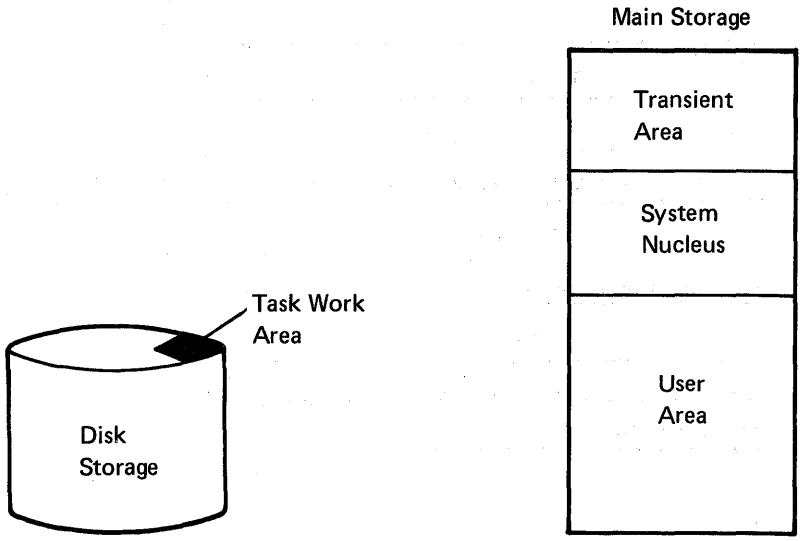
Chapter 2. System/34 Concepts

This chapter describes concepts of System/34 that are important in a work station environment. Understanding these concepts can help you plan your design more confidently and evaluate the design considerations presented in Chapter 3. The following concepts are described in this chapter:

- *System/34 storage concepts:* Describes the storage areas used by the System/34.
- *System/34 job processing:* Describes how the SSP (System Support Program) processes jobs and job steps.
- *Dispatching and swapping:* Defines dispatching and swapping and explains how the priority capabilities of the system affect dispatching and swapping.
- *Work station data management:* Defines work station data management and explains how it works.
- *File concepts:* Describes important System/34 file concepts such as file sharing and sector protection.
- *Printer concepts:* Describes spooling and the differences between the system list function and the printer data management function.
- *Libraries:* Describes library members, library size, and active user libraries.
- *Menus:* Describes fixed-format and free-format menus.
- *Program attributes:* Describes single requestor terminal (SRT) programs, multiple requestor terminal (MRT) programs, never-ending programs (NEPs), and programs that release the requesting display station.
- *Security:* Describes the security functions provided by System/34.
- *Interactive Communications Feature (SSP-ICF):* Lists the communications interfaces supported by SSP-ICF, describes the concept of sessions, and briefly describes the SSP-ICF data management function.

System/34 Storage Concepts

Understanding the storage areas used by the System/34 helps you use the system efficiently. The following diagram illustrates the important storage areas of the System/34.



DISK STORAGE

Disk storage is used by the system as a storage place for both programs as well as files and libraries.

Task Work Area

Programs are moved from main storage to disk and from disk to main storage. After they are removed from main storage, programs or portions of programs are stored in the task work area (TWA) of the disk. Also contained in the TWA are work areas used by active programs and control information used by the system.

The size of the TWA depends upon:

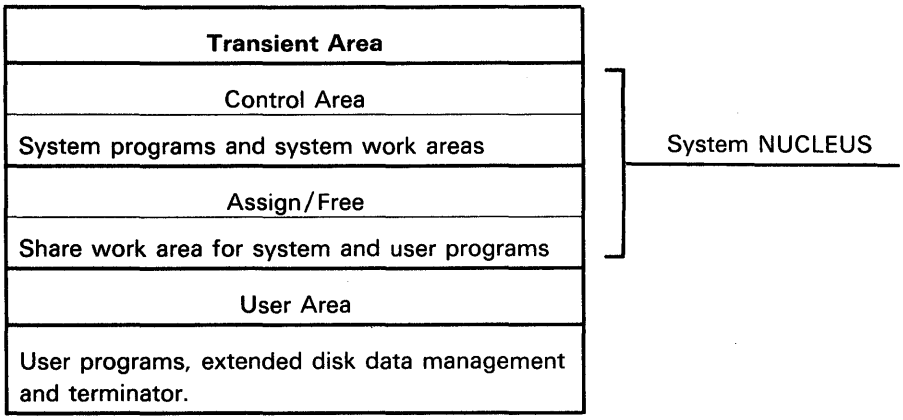
- Number of display stations being used
- Number of programs running within the system

If a program tries to acquire more disk space than is available within the TWA, the system attempts to allocate disk storage space. If there is enough additional space, the system expands the TWA, although this reduces the amount of space available for your files and libraries. An active program waits until the necessary amount of disk space is available when there is not enough room to expand the task work area.

MAIN STORAGE

Main storage contains programs, data, or instructions to the computer. It also contains work areas used by both application programs and the system.

The following diagram shows the basic parts of main storage.



Transient Area

System programs not required to be in main storage all of the time are called transient programs. There is a 2 K area of main storage called the transient area which is used to contain these programs. Transients are loaded into the transient area from the system library.

System Nucleus

The system nucleus manages system resources such as:

- Disks
- Printers
- Display stations

There are two main areas of the system nucleus:

- Control area
- Assign/Free area

Control Area

The control area contains the programs and work areas used by the system. Some of the items within the control area are:

- The SSP
- Work space used by the spool writer and spool intercept routine
- Work space used by the work stations
- The system routines that control your libraries and files on disk
- The system routines that control work station operations

Assign/Free Area

The assign/free area contains temporary storage space for both user and system programs. The more programs you have executing the more the system attempts to increase the size of the assign/free area by reducing the size of the user area. The system issues a message when the assign/free area can no longer be expanded.

Assign/Free Size

The maximum size of the assign/free area is 32 K bytes. The size of the assign/free area is affected by:

- Number of active tasks
- Number of never ending programs (NEPs)
- Number of multiple requestor tasks (MRTs)
- Number of work stations varied on or signed on
- Number of open files

Nucleus Size

The size of the nucleus you create on your system is important in three ways:

- The smaller your nucleus, the more main storage you can use for your programs.
- The larger your nucleus, the more system resource you can use at the same time. A resource is either a program, a file, an NEP/MRT, and so on.
- The system can potentially provide better response time and throughput as the nucleus increases in size.

  You want to have a nucleus size that gives you the maximum amount of main storage while obtaining the most work possible from the System/34.

The maximum nucleus size is 50K bytes.

User Area

System/34 loads your programs into the user area of main storage before running them. The user area also contains extended disk data management.

For more information about the nucleus size and performance, refer to Chapter 12 of the *Planning Guide*.

System/34 Job Processing

A basic understanding of the steps in System/34 job processing can help you design and code applications that more efficiently use System/34. This section defines the differences between jobs and job steps and describes the major functions performed during job processing. The following diagram shows the major SSP functions that are used to process your job.

JOBS AND JOB STEPS ON SYSTEM/34

On System/34, a job is a unit of work initiated by an operator at a display station or by a remote program that communicates with System/34 via the Interactive Communications Feature (SSP-ICF). One or more programs can be executed as part of a job. The execution of each program within a job is called a job step. Any of the following methods can be used to start a job on System/34:

- Entering OCL statements from the keyboard. When OCL statements are entered from the keyboard, the execution of a single program is considered to be a job. The LOAD and RUN statements, any OCL statements entered between them, and any utility control statements are processed as part of the job. OCL statements that are not entered between LOAD and RUN statements are processed as individual jobs. For example, the setting of switches via a SWITCH OCL statement that is not entered between a LOAD statement and a RUN statement is a job on System/34.

- Running a procedure. The operator can cause a procedure to be run by:
  - Entering a procedure command from the keyboard
  - Selecting an item from a menu
  - Placing the procedure on the input job queue

  The execution of the procedure requested by the operator is a job on System/34. If the requested procedure runs other procedures, those procedures are part of the job.

- Using the EVOKE OCL statement to evoke a different job from within a procedure.

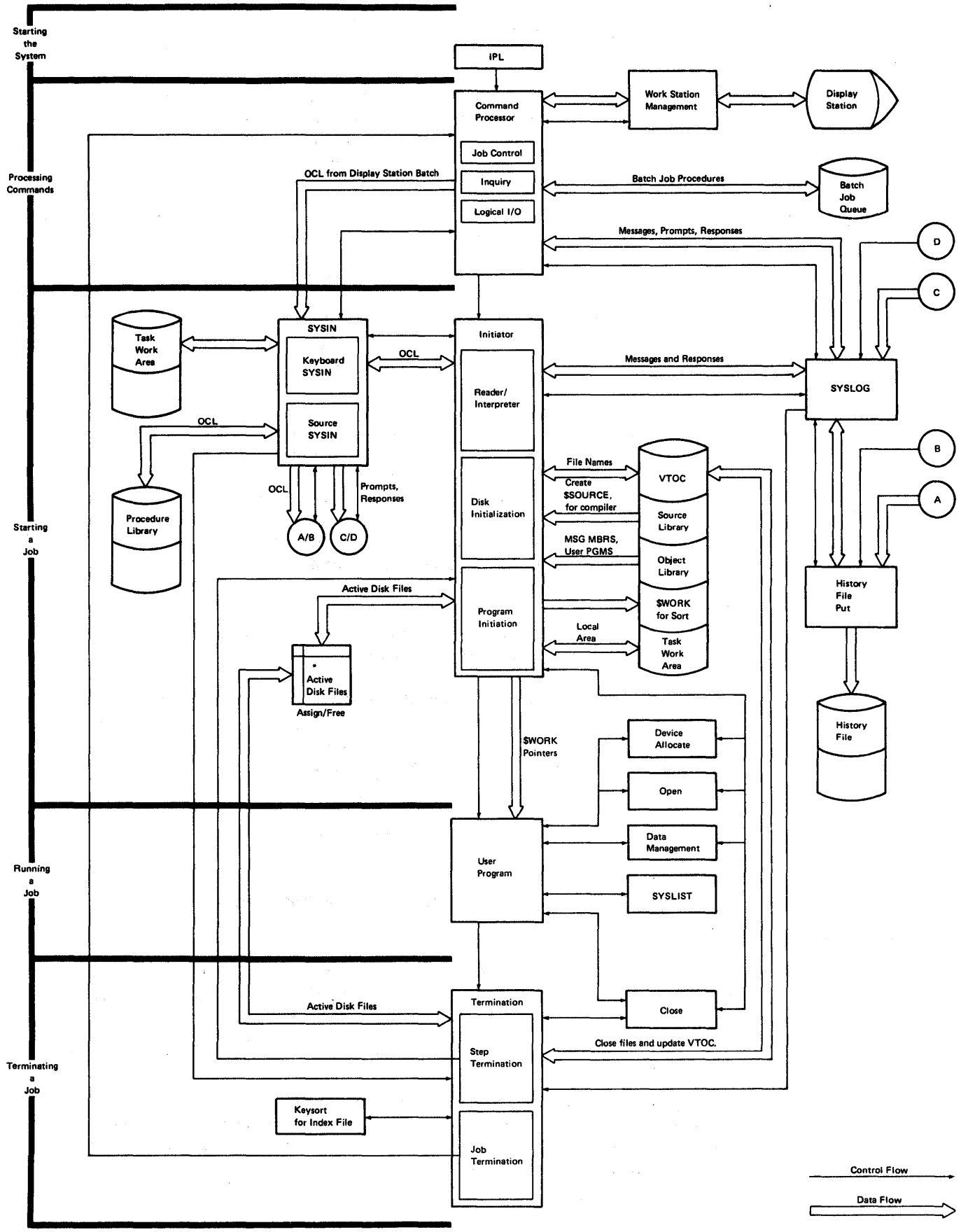
- Evoking the job from a remote program that is communicating with System/34 via SSP-ICF.

The SSP (System Support Program) assigns a unique job name to each job that the operator submits. The SSP-assigned job name has the following format:

wwhhmmss

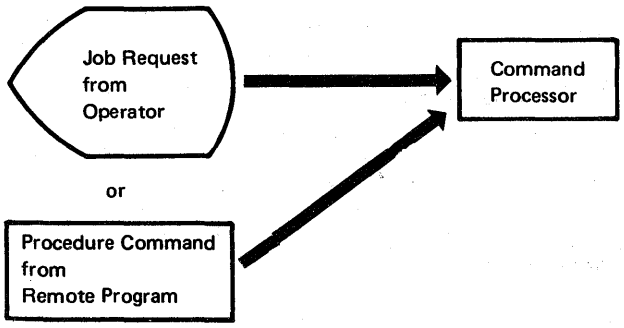
where ww is the work station ID of the requesting display station or the session ID of the associated SSP-ICF session, and hhmmss is the time the job was submitted in hours, minutes, and seconds based on the 24-hour clock, which is set by the system operator.

System Control Flow Overview

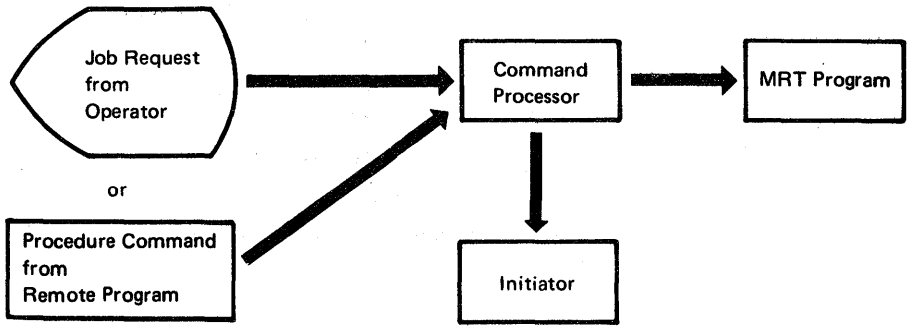


FUNCTIONS PERFORMED DURING JOB PROCESSING

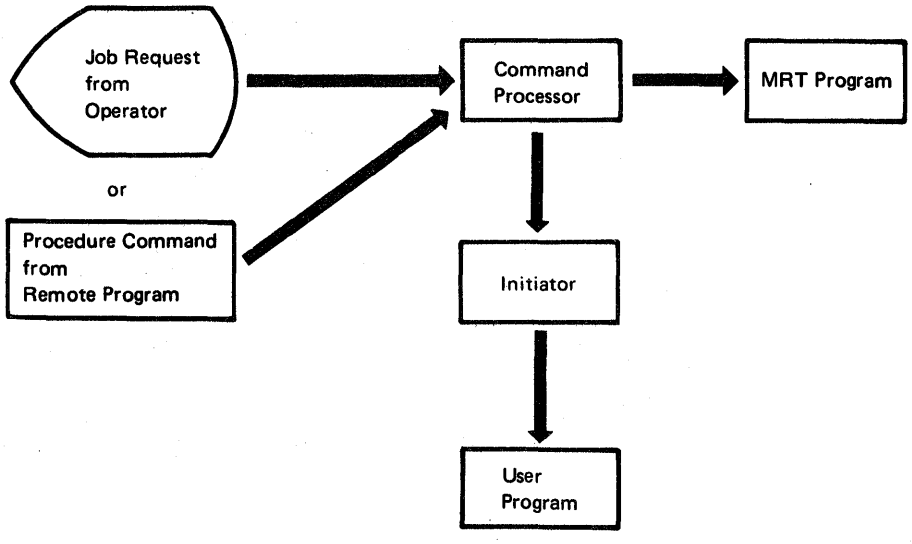
When the operator enters a statement or command on a command display or when he selects an item from a menu, the SSP function called the *command processor* processes the entry. If a remote program requests a job to be run, the command processor also processes the procedure command.



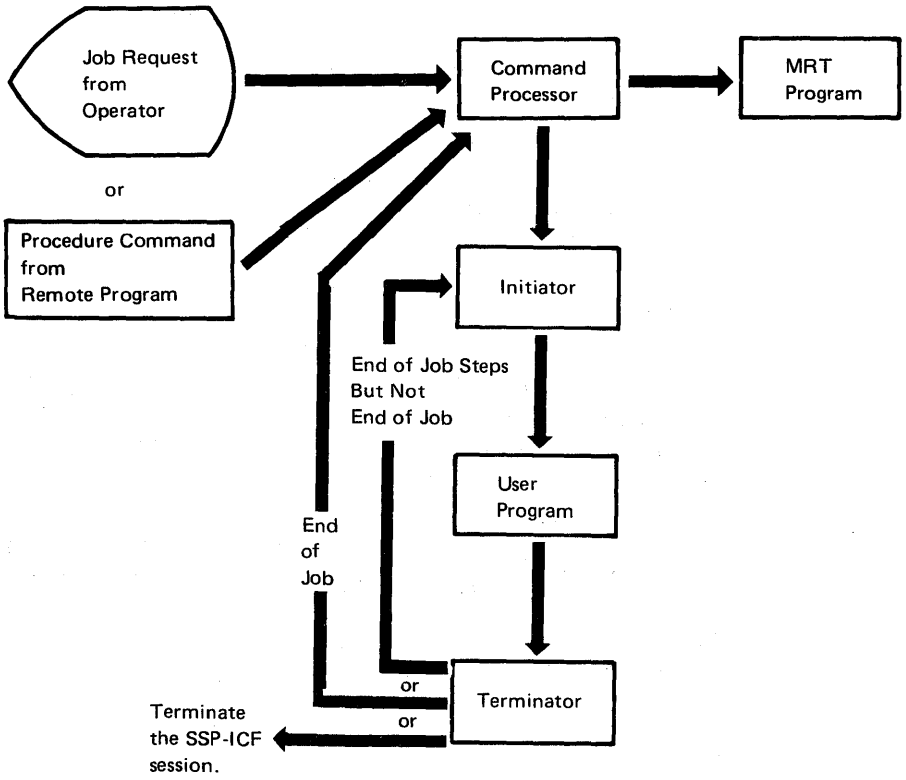
If the statement entered is a control command, the command processor performs the requested function. If the statement entered is not a control command, the command processor either passes control to the *initiator* function of the SSP or attaches the display station to an active MRT program if the procedure command is for an already active MRT program.



The initiator reads and processes OCL statements for the job. When it processes a RUN OCL statement, the initiator loads the requested program and passes control to it. The RUN statement is the last OCL statement in a job step.



When the step ends, the SSP *terminator* function performs system actions necessary to end the step. If more job steps follow, the terminator returns control to the initiator. If no other job steps follow, the terminator ends the job and either returns control to the command processor for locally initiated jobs or terminates the SSP-ICF session for remotely initiated jobs.



The following paragraphs provide more detailed information about the functions performed by the command processor, the initiator, the user program, and the terminator.

Command Processor

The command processor is the SSP function that initially processes information that the operator enters. When (1) the operator enters a command or selects an item from the menu, or (2) a remote program sends a procedure command request via SSP-ICF, the command processor checks the command entered or checks the statement associated with the selected menu item to determine whether a job should be initiated.

If the associated statement is an operator control command, such as the STATUS control command, the command processor does not initiate a new job. Instead, the command processor gives control to the SSP routines that immediately execute the control command.

If the associated statement is not an operator control command, the command processor next checks to see if the procedure command is a request for a currently active MRT program. If it is, the command processor attaches the display station or SSP-ICF session to the active MRT program. If it is not a request for an active MRT program, the command processor passes the statement to the initiator.

Initiator

The initiator uses the SSP system input function to read and process OCL statements from the system input device, which can be either the keyboard at the display station or procedure members in a library. During OCL processing, the initiator checks each OCL statement for valid parameters. The initiator function contains a routine for processing each of the OCL statements. These routines are loaded and executed when required by the initiator.

Functions provided as part of the initiator function include:

- Processing substitution expressions and condition tests. (This function is performed by the SSP system input function.)
- Checking the syntax of each OCL statement.
- Ensuring that required load members exist in the active user library or the system library.
- Ensuring that required files exist and are compatible with the parameters on the FILE OCL statement.
- Ensuring that required source and work files are available.
- Acquiring display stations for which REQD-YES is specified on the WORKSTN OCL statement.
- Releasing the requesting device if RELEASE-YES is specified on the ATTR OCL statement.
- Ensuring that the available user main storage is at least as large as the job region size. User storage occupied by nonswappable programs is not available.
- Allocating the SSP work areas required for the job.

If the processed procedure does not request an active MRT procedure via an INCLUDE OCL statement, the initiator loads the requested program and passes control to it.

The initiator executes in the user area of main storage at the same priority level as the program that is using the initiator.

If the initiator processes a procedure call for an MRT procedure, the actions performed by the initiator depend upon whether or not the MRT program is already executing. If the MRT program is already executing, the initiator attaches the display station to the executing program; if the maximum number of requestors is already attached, the initiator queues the display station to the program. If the MRT program is not already active, the initiator processes the statements (up to and including the RUN statement) in the MRT procedure, loads the MRT program, attaches the display station to the program, and passes control to the MRT program. When the MRT program releases the display station, the initiator regains control and returns to the calling procedure. For further information about MRT procedures and programs, refer to *MRT (Multiple Requestor Terminal) Program* later in this chapter.

The suggestions listed under *OCL Performance Considerations*, which follows, should help minimize OCL processing time.

OCL Performance Considerations

Minimizing OCL statement processing time is a good practice because excessive processing of these statements can increase job initiation time. The following suggestions should help minimize OCL statement processing time:

- Use defaults whenever possible. For example, code:

```
// FILE NAME-INVTY
```

not

```
// FILE NAME-INVTY,UNIT-F1,LABEL-INVTY
```

- Avoid using comments on OCL statements.

- Group OCL statements. For example, all WORKSTN statements should be grouped together. Grouping statements allows the system to load individual OCL processing routines once instead of loading them many times.

- Avoid continuation lines. For example, code:

```
// FILE NAME-DISKIP,DISP-SHR,LABEL-ORDERS
```

not

```
// FILE NAME-DISKIP,
//      DISP-SHR,
//      LABEL-ORDERS
```

- Use the local data area only when necessary. Processing information in the local data area requires additional disk accesses and can significantly increase OCL statement processing time.

- Use external switches rather than the local data area to condition the execution of steps in a procedure.

- When a sort is executed after it is tested, use the 3 print option, which prints only severe errors.

- After a procedure is tested, do not log OCL statements to the history file. Logging requires that every OCL statement processed by the system be written to the history file. Therefore, logging can significantly increase OCL processing time.

- For interactive applications, avoid using the // * statement to inform the display station operator that a procedure has started. Instead, use the first format displayed by the program to present that information.

- Limit the number of conditional expressions that are processed within a procedure. The GOTO and TAG statements should help you limit the number of conditional expressions that are processed.

- Use the PROMPT OCL statement to prompt for data that will be passed to the program when it begins processing. Using the PROMPT statement in this way allows the operator to key data into the first display while the SSP processes the rest of the OCL statements for the job step.

- Use the PROMPT OCL statement to prompt for procedure parameters instead of using the // * statement to issue operator messages and then prompting for input with ?nR? or ?nR'mic'? expressions. The // * statement and substitution expressions require more system activity than that required to display prompts and input fields with a display screen format. For example, a procedure displays selected records from a selected file. If the // * statement and substitution expressions are used to prompt for input, the procedure could be coded as follows:

```
// * 'ENTER FILE LABEL'
// IF ?1R?/ GOTO ERROR
// * 'ENTER STARTING RECORD NUMBER'
// IF ?2R?/ GOTO ERROR
// * 'ENTER ENDING RECORD NUMBER'
// IF ?3R?/ *'END OF FILE ASSUMED'
DISPLAY ?1?,,RECORD,?2?,?3?
// RETURN
// TAG ERROR
// PAUSE 'REQD PARAMETER--PROC CANCELLED'
```

When this procedure is executed, the operator is prompted for one parameter at a time. The operator presses the Enter/Rec Adv key after entering each parameter. The same procedure could be coded to use the PROMPT OCL statement. The procedure then could be coded as follows:

```
// TAG START
// PROMPT MEMBER-TRYFM,FORMAT-DISPLAY1
// IFF ?1?/ IFF ?2?/ GOTO GOOD
// PROMPT MEMBER-TRYFM,FORMAT-ERROR
// GOTO START
// TAG GOOD
DISPLAY ?1?,,RECORD,?2?,?3?
```

In this case, the display screen formats, DISPLAY1 and ERROR, are both in a load member called TRYFM. Figure 2-1 shows the S and D specifications for the two displays. (Instead of coding S and D specifications, you could use the Screen Design Aid (SDA) to create the display screen formats.) When DISPLAY1 is displayed, the operator is prompted for all three parameters. The operator enters all three parameters and presses the Enter/Rec Adv key. Figure 2-2 shows the display screen when DISPLAY1 is displayed.

Note: The check for null entries, which is coded in this example, would not be required if the fields were defined as mandatory-enter fields (Y in column 29 of the D specification).

IBM

System/34 Display Screen Format Specifications

Second Edition

Use this coding sheet only to define display screen formats for WSU and $SFGR. This coding sheet could contain typographical errors.

GX21-9253- U/M 050*
Printed in U.S.A.
*No. of sheets per pad may vary slightly.

S

Sequence Number	Form Type	Format Name	Format ID (WSU Only)	Start Line Number	Number of Lines to Clear	Lowercase	Return Input	Reset Keyboard (WSU Only)	Sound Alarm	Enable Function Keys	Enable Command Keys	Blink Cursor	Erase Input Fields	Override Fields	Suppress Input	Reserved	WSU Only: Enter Mode Sequence: Start	End	Entry Required	Repeat	Reserved	Priority	Preprocess	Review Mode Record Identifying Indicators: 1	2	3	Insert Mode Record Identifying Indicators: 1	2	3	Reserved	Key Mask	Reserved
1 2 3 4 5	6	7 8 9 10 11 12 13 14	15 16	17 18	19 20	21	22	23 24	25 26	27	28	29 30	31 32	33 34	35 36	37 38 39 40	41	42	43	44	45	46	47	48 49	50 51	52 53	54 55	56 57	58 59	60 61 62 63	64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	S	DISPLAY1																														

D

Sequence Number	Form Type	Field Name: WSU Field Name	Not Used by WSU	Field Length	Starting Location: Line Number	Horizontal Position	Output Data	Edit Code (WSU Only)	Input Allowed	Data Type	Mandatory Fill	Mandatory Entry	Self-Check	Adjust/Fill	Position Cursor	Enable Dup	Controlled Field Exit	Auto Record Advance	Protect Field	High Intensity	Blink Field	Nondisplay	Reverse Image	Underline	Column Separators	Reserved	Constant Type	Constant Data	Continuation
1 2 3 4 5	6	7 8 9 10 11 12	13 14	15 16 17 18	19 20	21 22	23 24	25 26	27	28	29	30	31	32 33	34	35	36	37 38	39 40	41 42 43 44	45 46	47	48	49	50 51 52 53 54 55	56	57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	D	FL0001		36	6	20	Y													Y							C	DISPLAY SELECTED RECORDX	
	D	S FROM A FILE																											
	D																												
	D	FL0002		20	9	8	Y																					CENTER FILE NAME --->	
	D	FILE		8	9	46	01	Y															Y		Y				
	D	FL0004		33	11	8	Y																					CENTER STARTING RECORD NX	
	D	UMBER --->																											
	D																												
	D	START		8	11	4	602	YN					Z										Y		Y			00000001	
	D	FL0006		31	13	8	Y																					CENTER ENDING RECORD NUMX	
	D	BER --->																											
	D																												
	D	END		8	13	4	603	YN					Z										Y		Y				
	D																												
	D																												

Figure 2-1 (Part 1 of 2). S and D Specifications for DISPLAY1 and ERROR

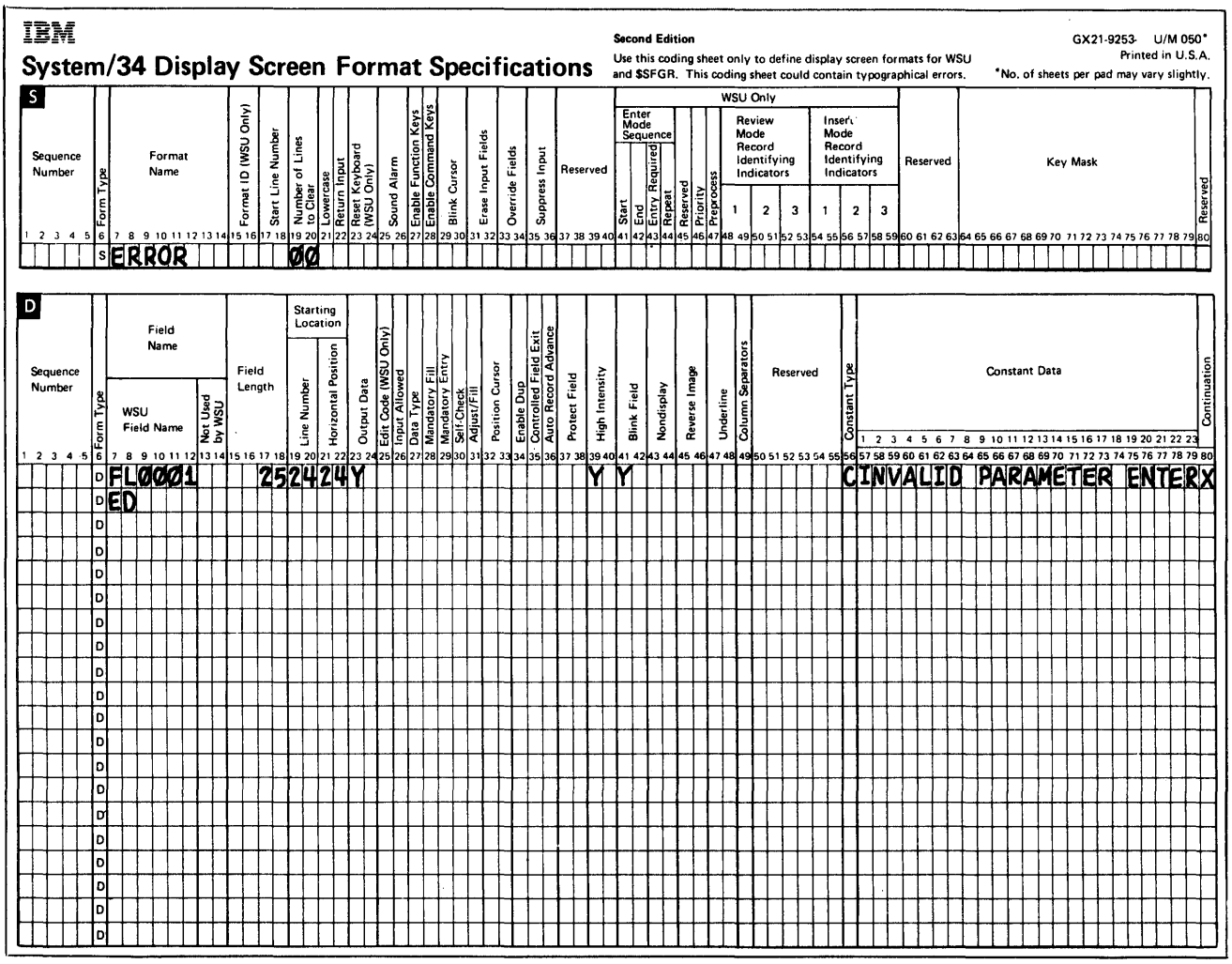


Figure 2-1 (Part 2 of 2). S and D Specifications for DISPLAY1 and ERROR

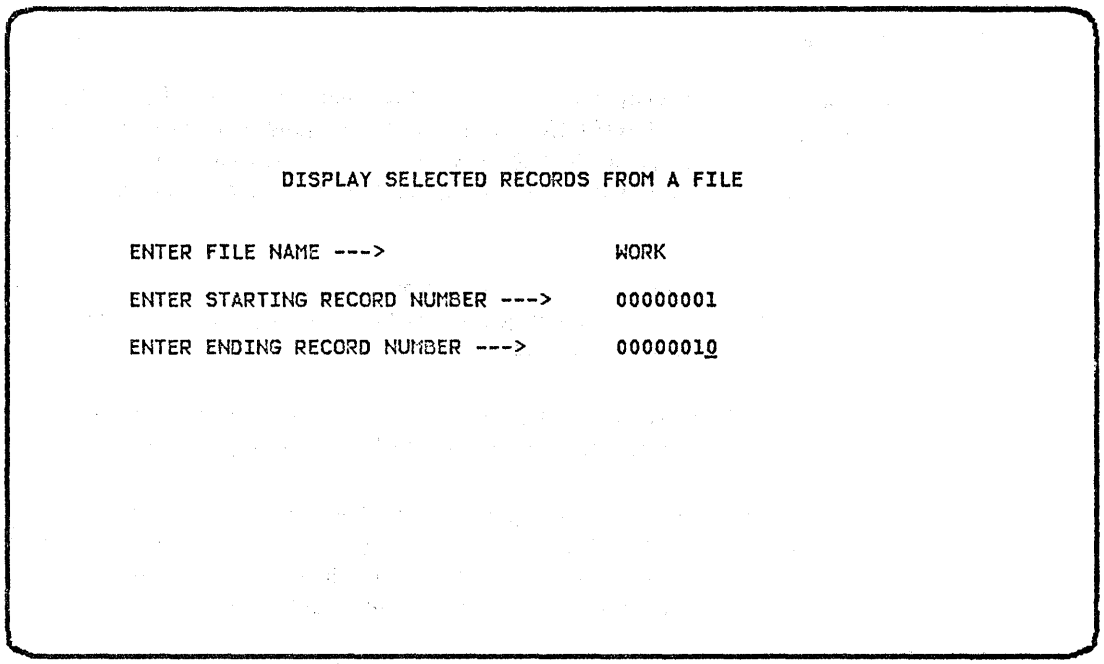


Figure 2-2. DISPLAY1 After the Operator Enters Parameters

User Program

Each program allocates and opens the files that it uses. In RPG II, WSU (work station utility), and FORTRAN programs, the programmer does not explicitly code these operations. After opening the files, the user program begins processing. Some of the services the SSP provides during execution of a user program are:

- Disk data management, which controls the flow of information to and from disk files. For information about disk files and disk file processing, refer to *File Concepts* later in this chapter.

- Printer data management, which controls the flow of information to the printer. For more information, refer to *Printer Concepts* later in this chapter.

- Optional spooling functions, which intercept printer commands and place them in disk storage, creating a spool file. When requested, the spooling function retrieves records from the spool file and prints them. For information about spooling, refer to *Printer Concepts* later in this chapter.

- Work station data management, which enables the user program to present data on a display screen by providing only a string of data fields and a format name. Work station data management then displays the data in the predefined format. For information about display screen formats and work station data management, refer to *Work Station Data Management* later in this chapter.

- SSP-ICF data management, which enables the program to communicate with programs on the same System/34 (INTRA subsystem) or with programs on another system. For more information, refer to *SSP-ICF Data Management* later in this chapter.

After the user program completes processing, it closes the files it used and passes control to the SSP terminator function. The RPG II, WSU, or FORTRAN programmer does not explicitly code these operations.

Terminator

Normal Termination

When the user program goes to end of job or when the operator selects option 2 in response to a displayed error message, the terminator function replaces the user program in main storage. The terminator performs the following steps:

- Updates the disk VTOC entries
- Frees system resources, such as main storage and assign free area (system queue space), that the program used
- Updates and initializes system data areas so that the SSP can initiate the next job step
- Terminates all previously acquired SSP-ICF sessions established by the program

If more job steps remain in the job, the terminator reloads the initiator so that the next step can be run.

If the step just ended is the last step in the job, the terminator also performs the following steps:

- Deletes all job files (RETAIN-J files) used by the job
- Deletes the reserve area that was requested for the job
- Releases the requesting display station if it is still attached to the job and returns control to the command processor so that the operator can request another job
- Terminates the requesting SSP-ICF session if the program was requested by a remote program and if the session is still active
- Frees the remaining system resources that were used by the job

Abnormal Termination

Abnormal termination of a program occurs when any of the following operator actions are taken:

- The operator selects option 3 in response to a displayed error message.
- The operator interrupts the executing program and selects option 2 or option 3 from the Inquiry display for all programs except MRT programs. For MRT programs, option 2 releases the display station from the MRT program and continues with the next job step; option 3 releases the display station from the MRT program and cancels the remaining job steps.
- The system operator uses the CANCEL control command to cancel the job.
- A program check occurs.
- The system detects an error condition during normal termination.
- The CANCEL keyword is executed in a procedure.

When an abnormal termination occurs for non-MRT programs, the terminator is loaded. Any remaining job steps in the job are not performed. If option 2 was selected from an Inquiry display, the files that were being used by the terminated job are closed. For all other types of abnormal termination, files are not closed, and the following statements are true:

- Files contain all updates made before the abnormal termination.
- Any additions made to nonshared files do not remain in the file unless the file is an IFILE.
- New files are not added to the disk VTOC.
- If keys were being sorted when the termination occurs, the file is marked as unusable. The index will be rebuilt by the IPL file rebuild function, which is described later in this chapter.

When the terminator function ends, it returns control to the command processor or terminates the SSP-ICF requestor session depending upon how the job was initiated.

The terminator executes in the user area of main storage.

SYSTEM INPUT (SYSIN) PROCESSING

The SYSIN (system input) function reads records from the input job stream, which is either entered from the keyboard or read from a procedure member. After reading a statement, SYSIN performs all substitutions and performs the functions specified by the statements that control SYSIN processing. The statements and expressions that control SYSIN processing are IF, IFT, IFF, RESET, ELSE, CANCEL, RETURN, END, GOTO, and TAG. For information about these statements and expressions, refer to Chapter 5 of the *SSP Reference Manual.*

After processing a statement, SYSIN gives the statement to the calling SSP function. During job initiation, the calling function is the initiator; therefore, all statements up to and including the RUN OCL statement are passed to the initiator. After a job is initiated, the statements are passed to the system utility program or the user program that requested system input processing.

The following example shows how SYSIN processes a typical statement. The example is intended to give a general idea of how SYSIN works, and is not intended to show the detailed logic of SYSIN processing. Before reading the example, you should be aware of the fundamental rules of system input processing:

- SYSIN processes a statement one field at a time from left to right. Fields are delimited by blanks.

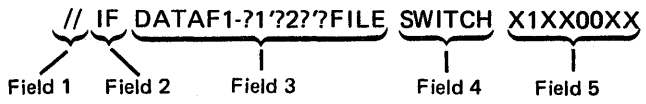
- Each time a substitution operation is performed, SYSIN goes back to the first field in the record and begins processing the record again. This must be done to allow for nested substitution.

- After all substitutions are performed, the length of the generated statement must be less than or equal to 120 characters. The actual length of the statement before substitution can be up to 240 characters.

- If substitution expressions follow the RUN OCL statement, job initiation time increases due to increased disk read and write operations required by SYSIN. You should use GOTO statements to make sure that conditional expressions and substitution expressions that follow the *RUN* statement are evaluated only if necessary.

System Input Processing Example

In this example, the following record is read from the input job stream:

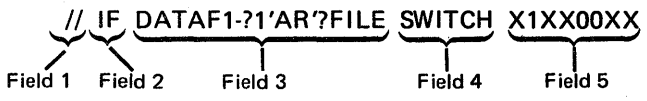


Assume that when this statement is read parameter 1 is undefined and parameter 2 has a value of AR. The system input function then performs the following steps:

Step 1. Identifies the first field as //b (b represents a blank).

Step 2. Identifies the second field as a valid system input expression (IF).

Step 3. Examines the third field and determines that the field contains a nested substitution expression. For a nested substitution expression, the innermost substitution is performed first. Therefore, SYSIN substitutes the value of parameter 2 (AR) into the expression. After the substitution, the record looks like this:



Because a substitution was performed, SYSIN goes back and begins processing the record at field 1.

Step 1. Identifies the first field as //b.

Step 2. Identifies the second field as a valid system input expression (IF).

Step 3. Examines the third field and determines that the field contains a substitution expression. SYSIN performs the substitution. In this case, parameter 1 is undefined and the value AR is substituted. The resulting record now looks like this:

Again, because a substitution was performed, SYSIN goes back and begins processing the record at field 1.

Step 1. Identifies the first field as //b.

Step 2. Identifies the second field as a valid system input expression (IF).

Step 3. Identifies the third field as an existence test. SYSIN performs the test and, in this case, determines it to be true.

Step 4. Evaluates the conditional expression formed by fields 2 and 3. The conditional expression specifies that because a disk file labeled ARFILE exists the remainder of the record should be processed. SYSIN discards the IF test (fields 2 and 3) and processes the remainder of the statement, which now looks like this:

After checking each field and determining that no further substitution or SYSIN processing of the statement is required, the statement is passed to the initiator.

Note: If the conditional expression is not satisfied, that is, ARFILE does not exist on F1, SYSIN discards the remainder of the record and reads the next record from the input stream.

Job Management and Job Scheduling on System/34

The IBM System/34 lets you assume an important role in the management and scheduling of your jobs. You can affect the order in which your jobs are presented to be executed by the use of different job queue priorities in the input job queue. You can affect the swapping and main storage processor utilization of your programs by the use of different execution priorities.

EXECUTION PRIORITIES

You can specify four different execution priorities for your job or job steps. These execution priorities may affect the swapping and the way your program gains control of the main storage processor from the dispatcher. The dispatcher is responsible for allocating the main storage processor to your program. The four execution priorities are:

- High
- Medium
- Normal
- Low

If you do not specify an execution priority for your job, the system assigns your job a normal priority.

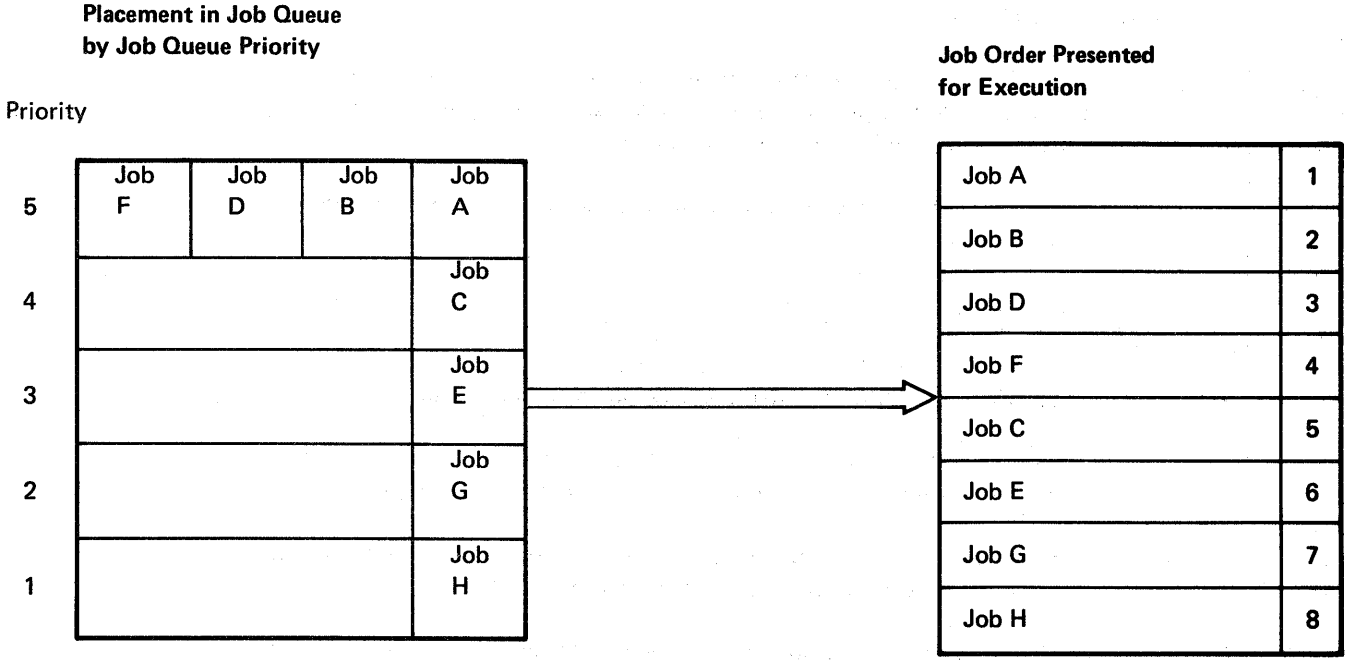
To specify the execution priority of your job(s), you can use the following:

- PRTY command
- // ATTR OCL statement

For further information about using the PRTY command and // ATTR OCL statement with execution priorities, refer to either the *SSP Reference Manual* or to the *Operator's Guide*.

Placement of Jobs on the Input Job Queue

The input job queue has five different priority levels numbered 1 to 5. Priority level 5 is the highest priority and priority level 1 is the lowest. By assigning job queue priority levels to your jobs, you can specify the placement of jobs on the input job queue and control the order in which your jobs are presented to the dispatcher to be executed. Jobs are placed on the input job queue on a first-in, first-out basis within job queue priority level. This means that all jobs with a level 5 job queue priority are presented for execution before jobs with a level 4 priority, and that before a priority 4 job can be considered for execution, all jobs with a job queue of 5 must have been dispatched. If you do not specify a job queue priority for a job that is using the input job queue, the system assigns your job a level 3 priority. The following illustration shows the order in which your jobs are presented for execution based upon placement in the input job queue.



Changing the Position of a Job Within the Input Job Queue

You can place a job in a higher or lower priority within the input job queue by using the CHANGE JOBQ command. The execution priority associated with the job is not changed.

Execution Priorities of Jobs in the Input Job Queue

The execution priority of a job placed in the input job queue is normal unless you use the PRTY command. If you use the PRTY command before placing a job in the input job queue, the execution priority is equal to the value specified on the PRTY command. If you use the PRTY command and do not specify a value, your job is assigned the high execution priority.

If you use a procedure to place a job in the input job queue, your job has the same execution priority as the procedure.

From the system console you can use the PRTY command to change the execution priority of a job in the input job queue. When you change the execution priority of a job, the job queue priority and its position within the job queue priority are not affected.

Initiating a Program

When a new program is to be initiated, the system arranges the programs by execution priority in the active program list and begins initiation of the programs with the highest execution priority.

The initiator on the system takes into account the following items regarding execution priority:

- Execution priority may be specified multiple times in a job by the use of the PRTY command or the // ATTR OCL statement. Each job step may have its own priority. The priority specified becomes effective as soon as the syntax of the OCL statements are validated by the initiator.
- When you have an NRT (no requestor terminal) program, the priority of the program is the same as the job that initiated the NRT program.
- When a job is started by the use of the // EVOKE OCL statement, the priority of the job evoked is the same as the job that evoked it.
- If you do not specify an execution priority for an MRT (multiple requestor terminal) program, the MRT program has the same priority as the job prior to the inclusion of the MRT program. If you attach a job to an MRT program, your job has the same priority as the MRT program. When a job is released from an MRT program, the job has the priority that was in effect prior to the inclusion of the MRT program.

DISPATCHING

Systems that allow only one program in main storage at a time waste considerable processor time. For example, when the executing program waits for an I/O operation, the processor is idle until the operation is complete. System/34, on the other hand, provides a dispatching function that allows multiple programs in main storage to share processing time. When a program that is using the main storage processor waits for the completion of an I/O operation or has executed for longer than a system-defined time limit, the system dispatcher gives control to another program in main storage that is ready to execute. The system dispatcher determines which program uses the main storage processor next.

To determine which program should use the processor next, the system maintains a list of programs that are in main storage and ready to execute. That list is called the *program ready list*. The dispatcher gives control to the program on the list with the highest execution priority. Programs are dispatched on a priority first-in, first-out basis. The following chart lists the dispatching sequence used by the System/34.

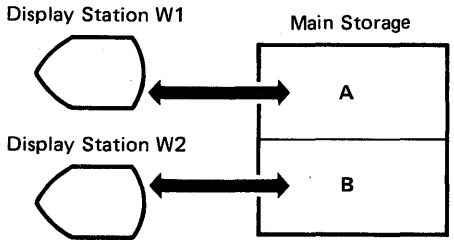
Execution Priority	Dispatching Sequence
System	1
High	2
Medium	3
Normal	3
Medium-low	4
Low	5

Note: The medium-low priority and system priorities are determined by the SSP and cannot be specified by the user.

At certain times, the system assigns a priority other than what you have specified. This assignment is temporary and is used to accommodate special situations such as termination of a job.

SWAPPING

Even though dispatching enables two or more programs in main storage to share the main storage processor, processing time can still be wasted. For example, if a program is so large that no other program can fit in main storage, the time that program spends waiting for I/O operations is wasted. Even if two or more programs fit into main storage, considerable amounts of time can still be wasted. For example, assume that two operators are running programs that require those operators to enter information from their display stations and that together the two programs, A and B, occupy all the available user storage. While one of the programs waits for an operation such as a disk or display station operation to be completed, the System/34 dispatching function allows the other task to execute.



In this situation, the main storage processor is not used continuously. For example, after Program A requests input from display station W1, that program waits in storage while the operator enters the input. If both programs wait for input from display stations at the same time, the processing unit might be idle for several seconds.

To make better use of processing time, System/34 provides a swapping function that allows the total amount of user storage required by concurrently executing programs to exceed the amount of main storage available for user programs. During swapping, the system temporarily removes a program or a segment of a program from main storage when the program cannot continue to execute because it is waiting to use some resource on the system. The system saves this program or segment of the program on disk so that another program can use main storage.

Note: The system swaps the program only if the main storage area is needed by another program.

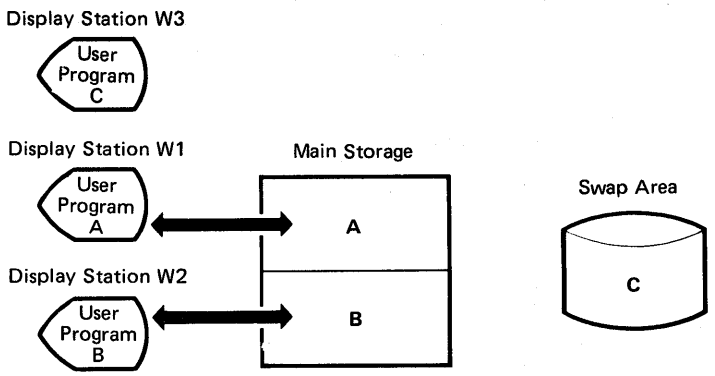
The swapping function uses the execution priority of a job to help it determine which jobs are to be swapped. The lower priority jobs in most cases are swapped first, leaving the higher priority jobs in main storage for as long as possible. The following chart lists the swapping sequence.

Execution Priority	Swapping Sequence
Low	1
Medium-low	2
Medium	3
Normal	3
High	4

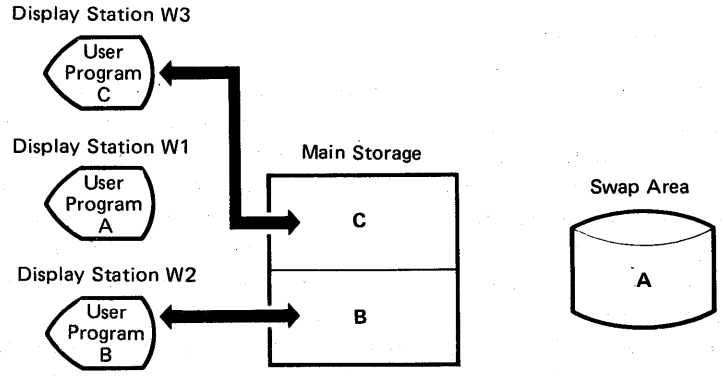
Ordinarily, swapping requires only fractions of a second. Most of the time, the operator at a display station is not even aware that the job is sharing the system with other jobs.

The following example shows how swapping can increase the total amount of work that concurrently executing programs can perform.

Assume that programs A, B, and C are executing concurrently and have the same execution priority. The programs are the same size. Programs A and B are in main storage; program C has been moved to the swap area on disk and is waiting to resume processing.



The system knows that program C is waiting to execute when program A requests input from display station W1. Program A will be inactive until the operator enters information. Therefore, the system transfers program A to the swap area on disk and transfers program C into main storage. Programs B and C share the main storage processor until program A is ready to be swapped back into main storage. In this example, the main storage processor will be idle only when all three programs are waiting for work station input.

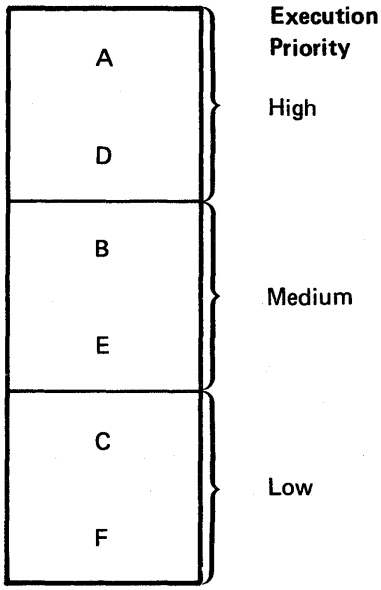


The following paragraphs further describe swapping on a general level, but they are not intended to be a complete discussion of the topic. The information presented, though not essential for successful use of System/34, provides a general understanding of how swapping works and how it can affect response time.

Active Program List

To keep track of the status of all active programs, the system maintains a list (or queue) of all active programs. An active program can be either in main storage or in the swap area on disk. The system uses the position of the program along with the execution priority to decide which programs to swap in and out of main storage.

Normally, programs on the active program list are in sequence by their execution priority. All high priority jobs precede medium priority jobs, and medium priority jobs precede low priority jobs. Low priority jobs are last on the active program list. For example, if programs A and D are high priority jobs, programs B and E are medium priority jobs, and programs C and F are low priority jobs, the list of active programs might look like this:

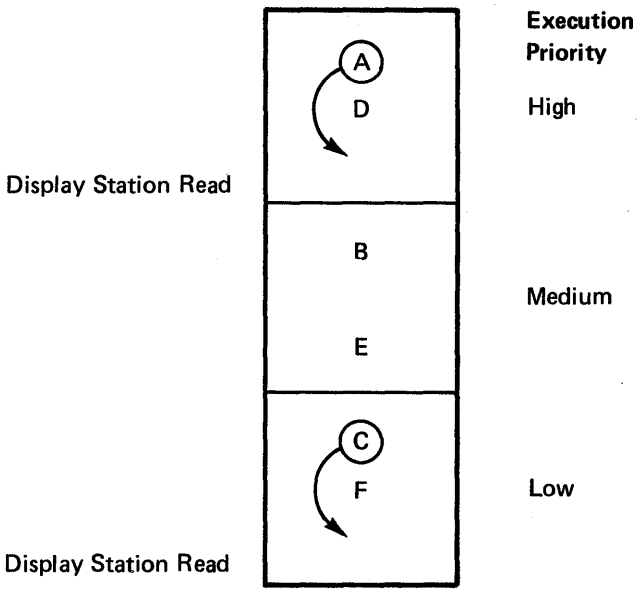


If you do not specify an execution priority or you specify normal execution priority for your job, the system assigns a normal execution priority to your job. The normal execution priority is considered equivalent to medium by the system. The system can change this system-assigned medium execution priority to a medium-low execution priority. This change occurs only when you do not specify an execution priority and your program executes for longer than a system-determined time limit without performing a display station read operation. The time limit, called the interactive time limit, is approximately (N+1) x 1/2 second (1/2 second=500 milliseconds), where N is the number of display stations attached to the program. The interactive time limit also deducts I/O processing time as well as main storage processor time.

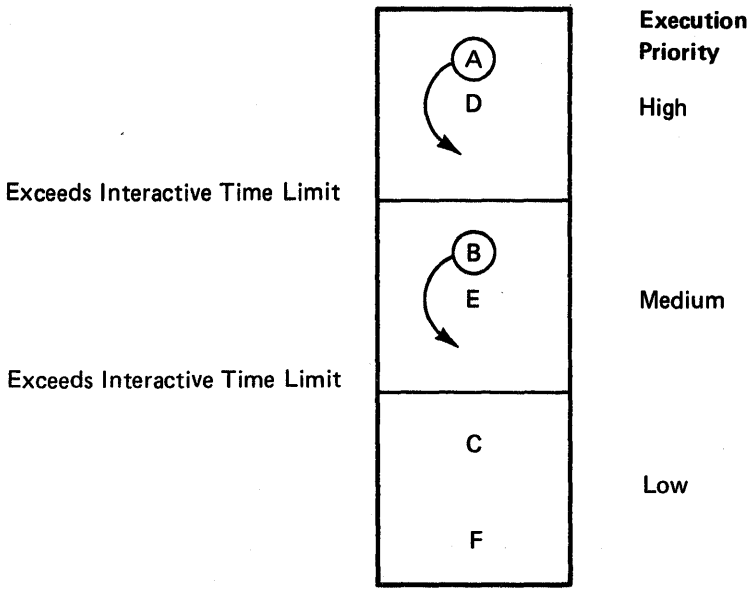
For more information about the interactive time limit, refer to the *System Measurement Facility Reference Manual.*

The following examples show how a program's position on the active program list can be changed, when you have specified an execution priority.

- When you specify an execution priority and your program performs a display station read operation, that program is moved to the bottom of the list of programs with the same priority.

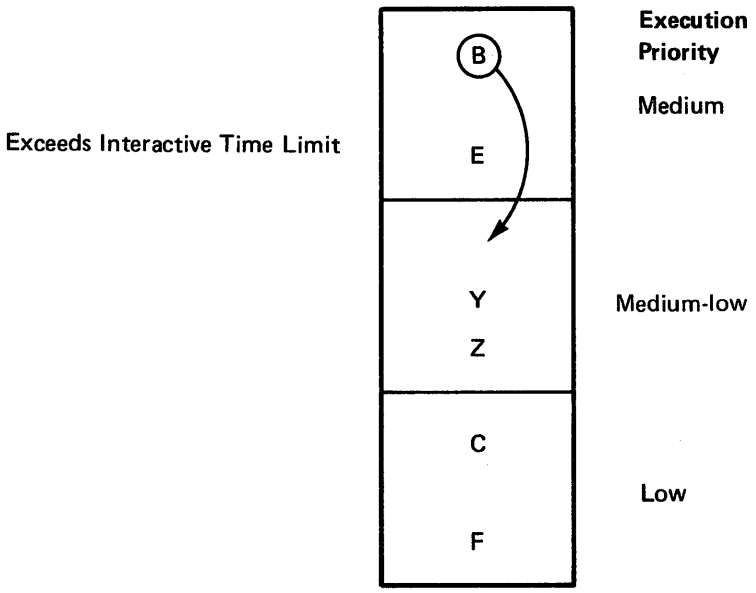


- When you specify an execution priority and your program exceeds the interactive time limit, that program is moved to the bottom of the list of programs with the same priority.

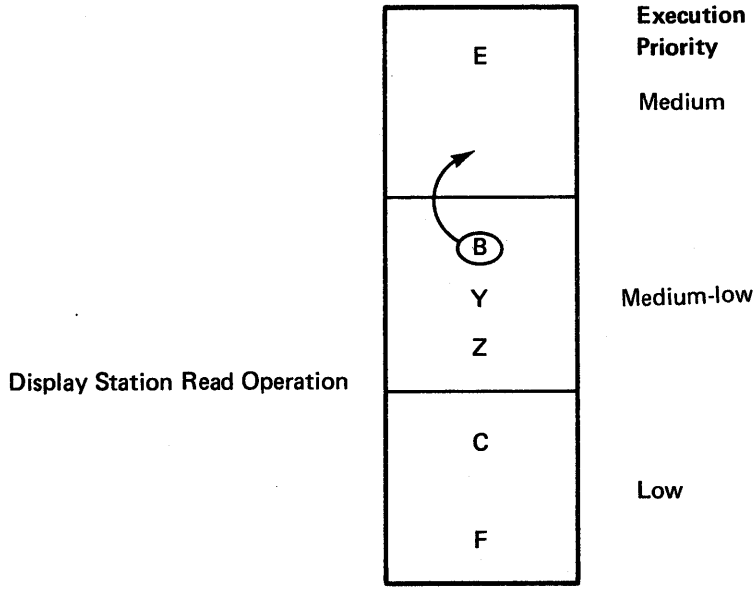


The following examples show how a program's position on the active program list changes when you do not specify an execution priority and you let the system assign normal priority to your program. You should be aware that the system initially considers normal priority equivalent to medium priority.

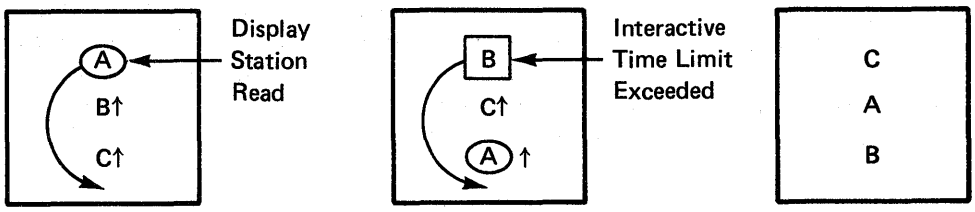
- When your program exceeds the interactive time limit, that program is moved to the top of a system defined list with medium-low priority. Only the system can specify a medium-low priority for your program.



Your program remains in this medium-low classification until a display station read operation is performed by your program. When your program performs a display station read operation, the program is moved by the system to the bottom of the list of the medium priority programs.



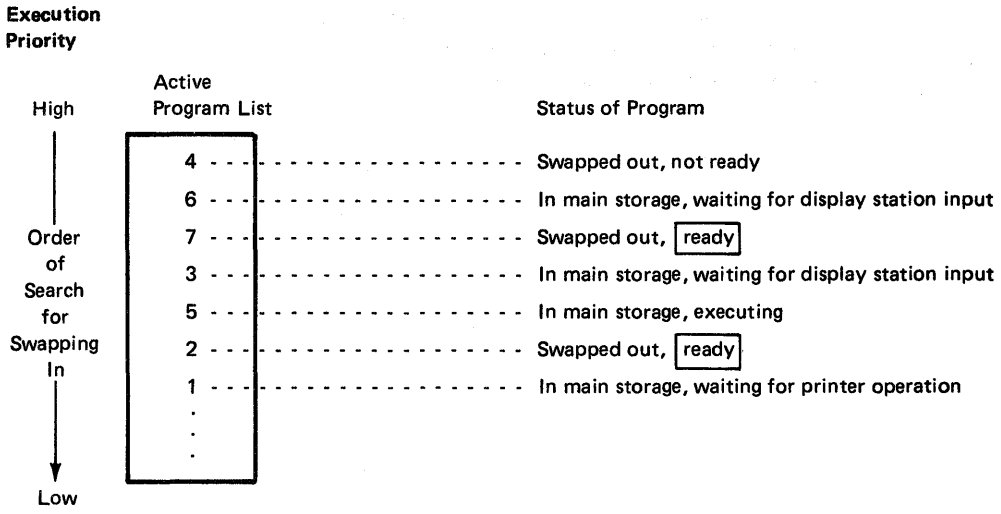
Your program remains at the bottom of the list of programs with the same priority until programs ahead of it perform display station read operations or exceed their interactive time limit. When these conditions happen, your program is moved up on the list. The following diagram shows the movement of program C on the active program list.



How the Swapping Function Uses the Active Program List

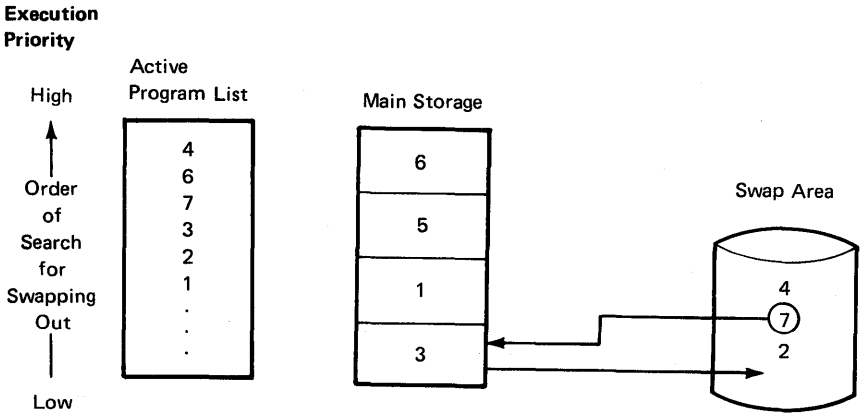
The System/34 swapping function is activated whenever the status of an active program changes such that swapping may occur. For example, swapping can occur when a program issues a display station read operation.

As the first step in swapping, the system starts at the top of the list of active programs and checks the status of the programs to determine whether a program is waiting to be swapped in. The first swapped-out program of the highest priority the system finds that is ready to swap back in is the program that will be swapped in. For example, suppose that seven programs are on the list. Of those programs, 2, 4, and 7 are swapped out, and 2 and 7 are ready to run. If the active program list looks like this:



then program 7 is the program that will be swapped in.

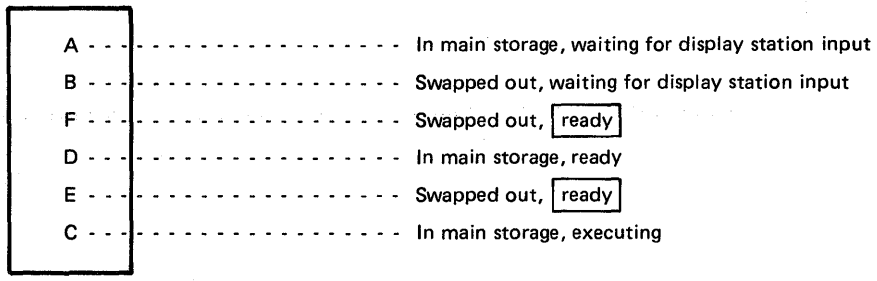
If a program is ready to swap in, the system must determine which program or programs to swap out. To determine which programs to swap out, the system starts at the bottom of the active program list and searches for programs in main storage that are waiting for input from a display station. The system swaps out the first program it finds that satisfies the search and swaps in the other program if enough storage is made available. The system might have to swap out two or more programs to free enough main storage for the program to be swapped in. In the preceding example, program 7 is to be swapped in. As shown in the example, programs 3 and 6 are both waiting for display station input. The system will swap out program 3, which has the lowest priority on the list, and swap in program 7.



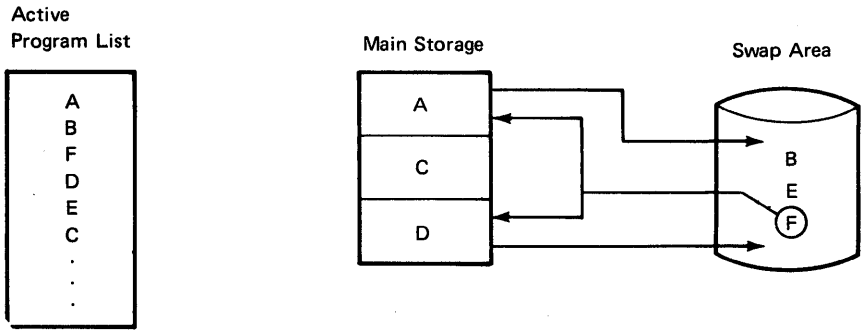
If swapping out program 3 does not free up enough space for program 7, the system continues searching until it finds another program or programs to swap out along with program 3. In this example, program 6 is also waiting for display station input and could be swapped out along with program 3. Program 7 could then be swapped in.

If the system does not find any programs waiting for display station input or if the amount of storage used by the programs and the amount of unused storage is not large enough to contain the program to be swapped in, the system goes back to the bottom of the active program list and searches for programs that have been executing in main storage for longer than 1/2 second, including I/O time. Such a program can be swapped out only if it is a lower priority on the active program list than the program to be swapped in. By allowing a program to execute for at least 1/2 second, the system avoids constantly swapping a program in and out while the program uses little or no processor time. For example, assume the following:

- Programs A, C, and D are in main storage.
- Of the programs in main storage, program A is waiting for display station input, and program D had been executing for longer than 1/2 second.
- Programs B, E, and F are swapped out.
- Of the swapped-out programs, programs E and F are ready to be swapped in.
- The active program list is as follows:



The system begins with the highest priority program at the top of the active program list and searches for a swapped out program that is ready to resume processing. In this example, the first such program that the system finds is program F. Therefore, program F is the program that will be swapped in. Now, the system begins searching for a program or programs to be swapped out to make room for program F. The system begins at the bottom of the active program list with the lowest priority program and searches for programs in main storage that are waiting for display station input. Program A is the only such program and will be swapped out. If enough space still does not exist for program F, the system returns to the bottom of the list and searches for programs that have been executing for longer than 1/2 second. The first such program the system finds is program D. Because program D is of lower priority than program F on the active program list, program D can also be swapped out to make room for program F.



The swapping and dispatching activity just described is controlled by programs that execute in control storage; therefore, the swapping and dispatching functions are executing at the same time as the programs that are using the main storage processor.

Swapping Times

The amount of time System/34 takes to swap a program or segment of a program is directly related to the size of the program. The System/34 swaps only as many 2 K byte segments of a program as is necessary. For example, a 12 K byte program might require about 90 milliseconds to swap in and about 130 milliseconds to swap out. By comparison, a 24 K byte program might require about 140 milliseconds to swap in and about 230 milliseconds to swap out. The actual swapping time will vary depending upon the disk drive being used and other disk activity within the system.

Job Priority

When assigning jobs to different priority levels within the input job queue and specifying different levels of execution priority, your main goal is to process the maximum number of jobs in the least amount of time.

You may want to use the input job queue and execution priorities to establish groups of jobs with certain characteristics. For example, you may want to assign all nonswappable jobs a specified job queue priority with a specific execution priority, so that jobs that use display stations have been executed before your nonswappable programs begin to execute. You may want to run your testing jobs with one execution priority and your production programs with a higher execution priority. You may want to assign execution priority to programs based upon the following criteria:

- Main storage size of the program. You may want to assign a lower execution priority to programs that use a larger amount of main storage.
- Whether the program can be run as an interactive or batch program. You may decide to assign higher execution priority to interactive programs than to batch programs, depending on your processing requirements.
- Amount of main storage processing time the program uses. You may decide that a program that uses a large amount of main storage processor time should have a lower execution priority than a program that does not use as much main storage processor time.
- How much elapsed time it takes the program to run. You may want to assign a high execution priority for a job that does not take a long time to execute and complete its processing on the system.
- What kinds of demands the program makes on system resources, such as disk files. You may want to base your assignment of execution priority to a job based on the total demands it makes on the system. For example, a program that uses four files and a printer may have a lower priority assigned than a program that uses only one file and does not use a printer.
- How important the program is to your data processing requirements and deadline schedules. You may want programs that are extremely important to your organization assigned a high priority every time they are run.

You may also let the system set the job queue and execution priorities for you.

Execution Priority Hints

In general, let the system assign execution priorities to your job(s): the normal defaults can usually handle your job needs. However, you should be aware of the following when setting your own execution priorities:

- Assign high priority to jobs requiring either fast response time at a display station or quick throughput.
- Assign low priority to batch jobs that will run for a long time.
- Assign medium priority to interactive jobs that could be reclassified as batch jobs by the system, based upon the processing requirements of the program.

Nonswappable Programs

When you run the overlay linkage editor, you can specify that a FORTRAN, COBOL, or assembler program is nonswappable. You cannot define an RPG II, BASIC, or WSU program as nonswappable. A nonswappable program, once it is initiated, remains in storage until the program completes. Because a nonswappable program decreases the amount of main storage available to other programs, the amount of swapping usually increases when a nonswappable program is being run. Therefore, the performance of other jobs can be degraded when a nonswappable program is run. Because the performance of all other programs may be adversely affected and the capability of the system to initiate new jobs restricted, the nonswappable attribute should be used only when absolutely necessary.

When a nonswappable program is run, the amount of main storage available for swappable programs is diminished. The amount of storage used by a nonswappable program is called *nonswappable storage*. The remaining user storage is called *swappable storage*.

Work Station Data Management

User programs that communicate with display stations use a system function called work station data management to write to and read from a display station. A *display screen format* defines to work station data management what information should be displayed and read from the display station. When a program requests an operation, the program supplies to work station data management the name of the format that defines the display. Formats used by a program are kept in a library load member separate from the program. The formats do not have to be kept in the library that contains the program. One display format load member can contain up to 32 formats. For an RPG II program, the default name of the display format load member consists of the program name followed by the characters FM. For example, if the RPG II program is called INV, the default display format load member is named INVFM. The FMTS continuation option on the file description specification allows you to specify a different name for the format load member or *NONE if only interactive communications formats are used.

Display screen formats are generated automatically by some programs or must be defined by programmers for other programs. Examples of automatically generated formats are those used by DFU (Data File Utility) and the RPG II CONSOLE file. Examples of formats that must be explicitly defined by the programmer are the formats used with RPG II WORKSTN files and the formats used by WSU, COBOL, FORTRAN, and BASIC programs.

If you are programming in RPG II, screen formats do not have to be defined as the program name followed by the characters FM. You can also use the same format member in more than one program.

The programmer defines display screen formats by:

- Using SDA (Screen Design Aid) to interactively define the formats. Detailed information about SDA is in the *SDA Reference Manual.*

- Creating S and D specifications. The programmer can use SEU to place these specifications in a source member. For programs other than WSU programs, these S and D specifications are used as input to the $SFGR SSP utility program, which generates the actual display screen format and places it in a library load member. For WSU programs, the S and D specifications are included as part of the specifications for the program. The WSU procedure calls $SFGR to process those specifications and generate the display screen format. For detailed information about the entries on the S and D specifications, refer to the *SSP Reference Manual* and the *WSU Reference Manual.*

To use the functions provided by work station data management, the programmer should know what kinds of information he supplies when he defines a display screen format and how that information controls the operations performed by work station data management. The following sections describe:

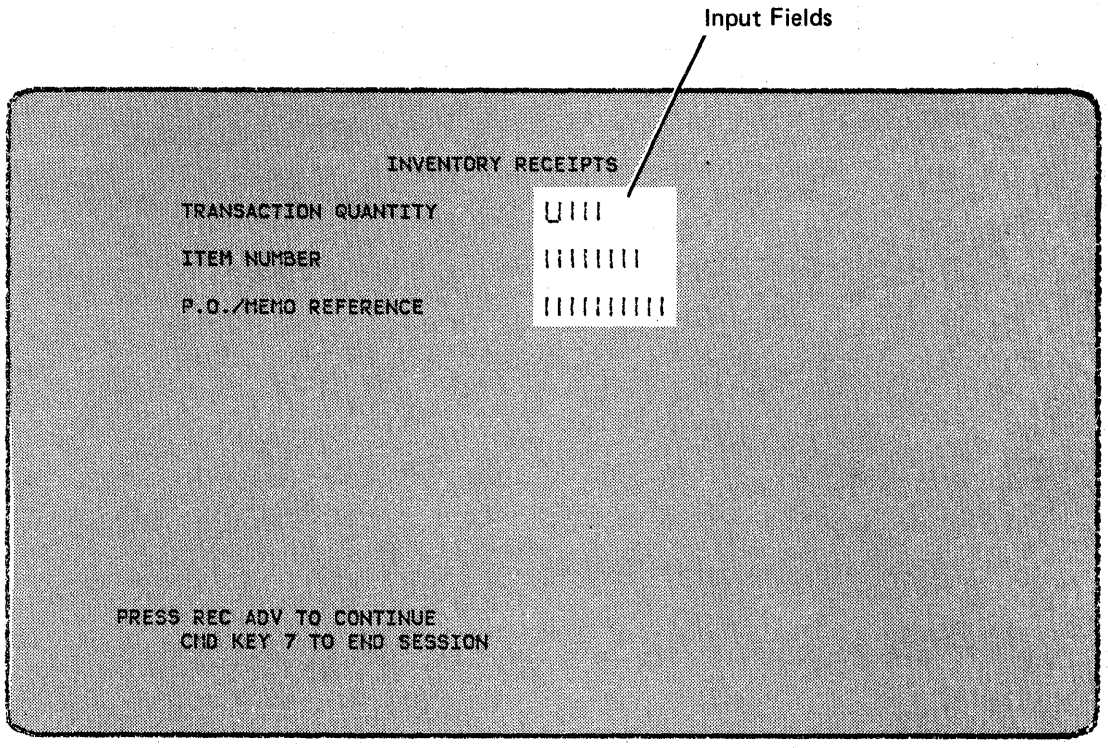
- How information supplied by the programmer affects individual data fields
- How information supplied by the programmer affects the operations performed by work station data management

Note: If you define a screen format that is larger than the screen size of the display station that you are using, an error condition occurs. For example, if your screen format is 1000 characters and you are using a 960-character screen size, an error condition occurs.

DATA FIELDS IN A DISPLAY SCREEN FORMAT

When the programmer defines a display screen format, he defines it in terms of the fields on the display. He specifies each field as an input field, an output field, or an output/input field.

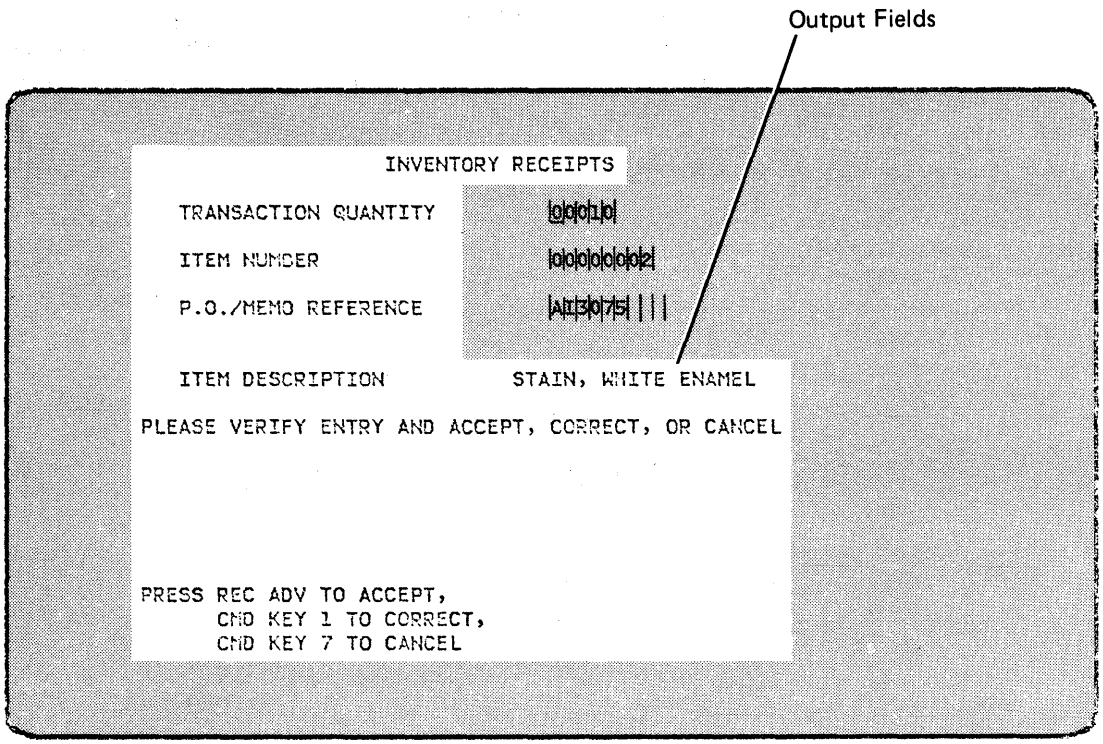
Input fields are fields into which the operator can enter data. When a program displays a format, input fields are normally blank. The contents of input fields are sent to the program when the operator presses (1) the Enter/Rec Adv key, (2) an enabled user command key, or (3) an enabled roll key or when the operator exits from a field for which auto-record-advance was specified.



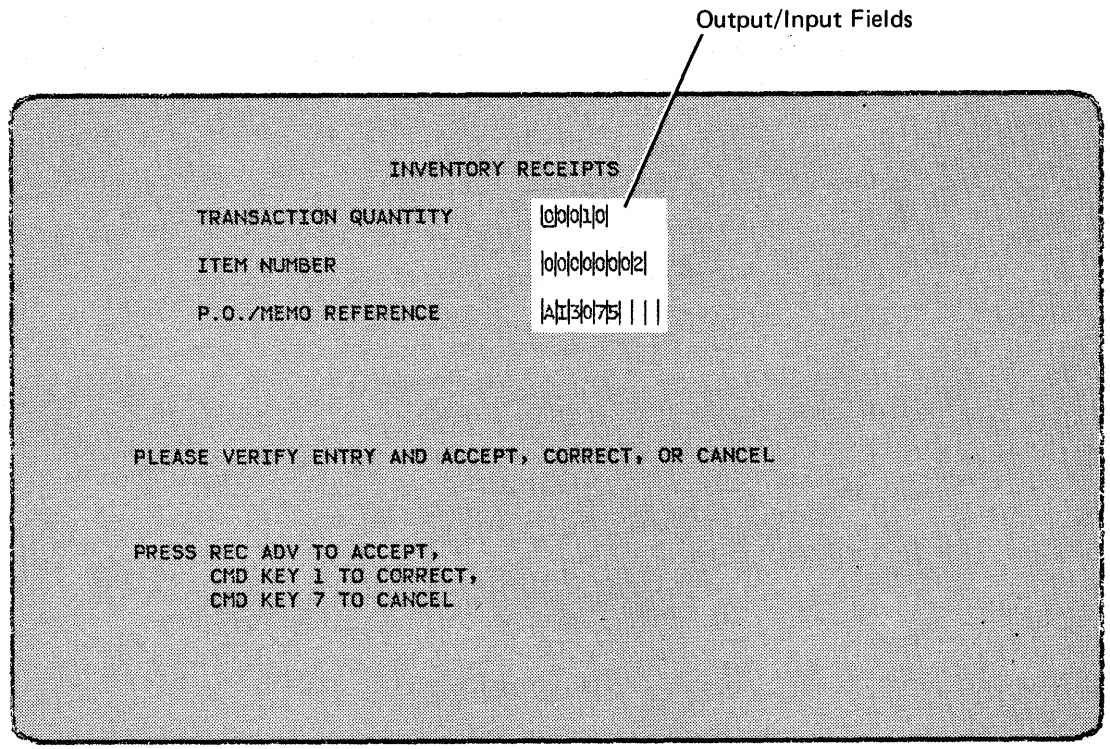
Note: The vertical lines in the input fields are column separators, which can be requested by the programmer when he defines the format.

Output fields contain data that the display station operator cannot change. The contents of output fields are not returned to the program when the operator enters the display.

The program can supply the information displayed in an output field, or the programmer can provide the information as part of the format definition. (Prompts and constant values specified when a format is created are output fields.)



Output/input fields contain information that is either supplied by the program when it displays a format or specified by the programmer when he defined the format. The display station operator may change the information in output/input fields. The information is returned to the program when the operator presses (1) the Enter/Rec Adv key, (2) an enabled user command key, or (3) an enabled roll key or when the operator exits from a field for which auto-record-advance was specified.



Field Attributes

In addition to defining field type, the programmer defines the location of the field on the display and the type of data that can be entered into the field (for example, alphameric data or only signed numeric data). The programmer also defines physical characteristics such as display intensity and whether or not the field is blinking. The description of the $SFGR utility program in the *SSP Reference Manual* explains each of the field characteristics that the programmer can define. Chapter 3, *Design Considerations* presents some considerations for designing displays.

WORK STATION DATA MANAGEMENT OPERATIONS

The Work Station Buffer

Work station data management uses an area in main storage called the work station buffer (or work station queue space) as a buffer for work station operations. For output operations, work station data management prepares a format for transmission by merging data supplied by the program with display control information from the format load member. The merged information is placed in the work station buffer. Work station data management then transmits the contents of the buffer to the display station. For input operations, work station data management normally uses the work station buffer as an input buffer for information received from the display station.

The size of the work station buffer is specified during system configuration. The size of the buffer can affect system performance. If the buffer is larger than necessary, the amount of storage available for user programs is decreased, and more swapping might take place. If, on the other hand, sufficient buffer space is not available for an operation, system performance might be degraded. The action taken by the system when the buffer space is not sufficient depends on whether the display station is a local or remote display station.

If a format is being transmitted to a local display station and if the size of the format exceeds the configured work station buffer size, work station data management writes a portion of the user program to the disk and uses the freed area as work station buffer space. After the information is transmitted, the saved portion of the user program is returned to main storage, and the program can resume processing. This activity affects system performance in two ways:

- Additional disk operations are required to perform the display station output operation. These additional operations increase the time taken to perform the output operation.

- The space occupied by the user task becomes nonswappable until the display station output operation is completed.

Normally, on a read operation from a local display station, the program is swapped into main storage (if the program has been swapped out) when the operator presses the Enter key. The SSP reads the information into the buffer at the same time that the program is being swapped in. However, if sufficient work station buffer space is not available, this overlapping of operations cannot occur. Instead, the program is swapped into main storage, and work station data management reads the data directly into the input area in the program.

If a format is being transmitted to a remote display station and if sufficient work station buffer space is not available for a display station output operation, work station data management does one of the following:

- Save on disk any input information in the work station buffer and use the freed space for the output operation.

- If enough space cannot be made available by saving input data, work station data management writes the data onto the disk and issues the output operation from the disk area.

When input is received from a remote display station, work station data management normally places the received data in the work station buffer. If sufficient work station buffer space is not available, work station data management writes the received data onto the disk. After the program is swapped into main storage, the data is transferred from the work station buffer or from the disk into the program's input/output area.

Normal Operations

Normally, when a program requests that work station data management send a display screen format, work station data management accepts the request and prepares the specified format for transmission by merging data supplied by the program and display control information from the format load member stored in the assign/free area of main storage. Work station data management places the merged information in the work station buffer. The user program can then resume execution without waiting for the format to be transmitted. After transmitting the format, work station data management invites input from the display station. The system will not accept input from the display station until work station data management invites input from the display station.

When the user program requests work station data management to read the information entered on the display, work station data management performs an accept input operation, which waits for information to be entered from an invited display station.

Input may have been invited from more than one display station attached to the program. When a display station operator enters data from an invited display station, work station data management causes the input data to be read into the user program's buffer area.

Figure 2-3 summarizes these steps.

Note: Work station data management does not invite input if you use the suppress input operation, which is described later in this section. If a program attempts to read from a display station from which input is not invited, the display station will no longer be able to communicate with the program.

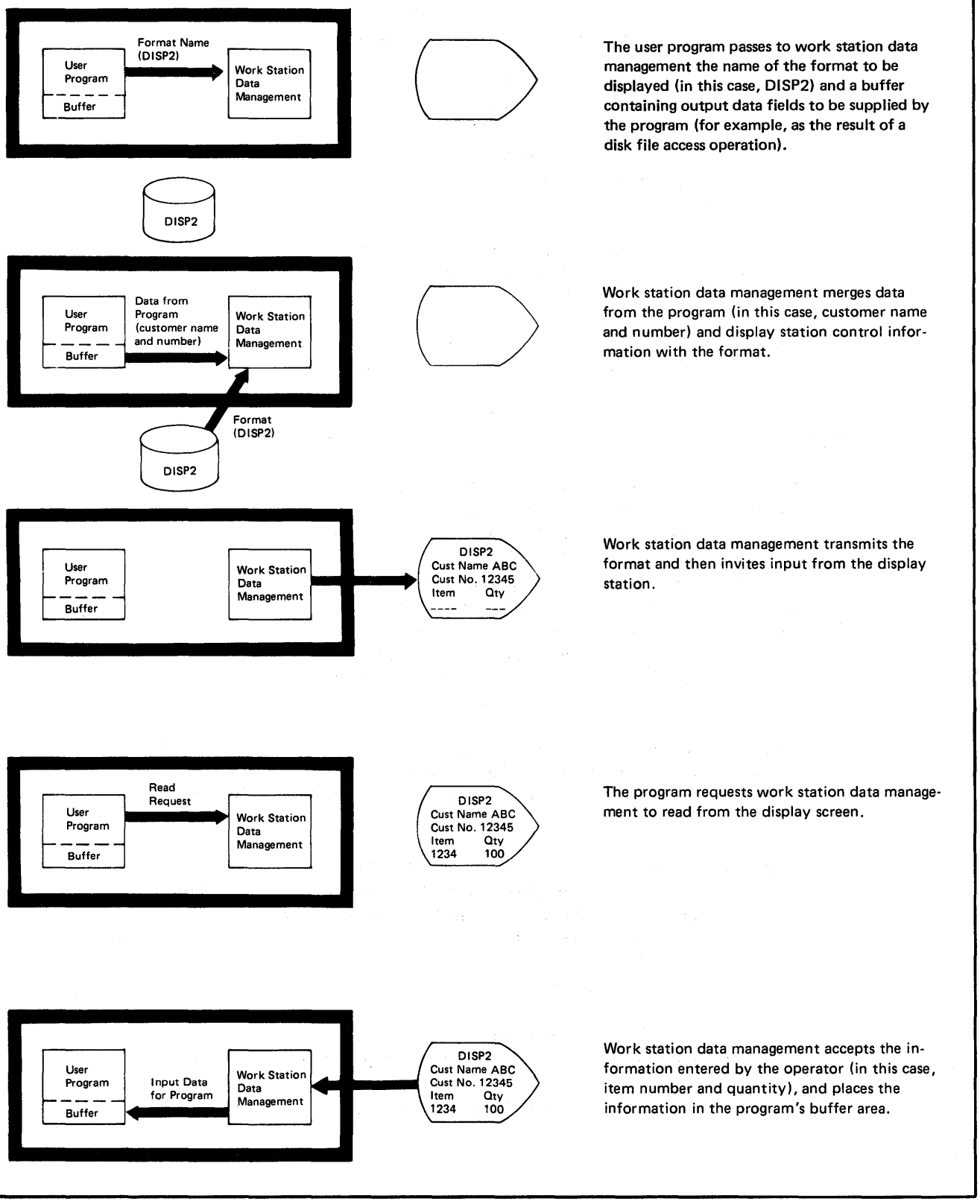


Figure 2-3. Steps in Normal Processing Using Work Station Data Management

Modified Operations

When a programmer defines a display screen format, he can modify the operations normally performed by work station data management, or he can identify indicators that conditionally modify the operations when the format is displayed. The following sections describe the modified operations that can be requested. Each section refers to the columns on the S specification that specify the operation. If the programmer uses SDA rather than the S specification, he requests these operations by responding to prompts.

Erase Input Fields Operation

For an erase input fields operation, which is specified in columns 31 and 32 of the S specification, work station data management blanks out the contents of the input and output/input fields on the display. Work station data management then invites input from the display. The format is not sent to the display station on an erase input fields operation. Therefore, the programmer might want to request the erase input fields operation when an application requires an operator to enter information on the same screen time after time. In such an application, the programmer should specify that the erase input fields operation be controlled by an indicator. The first time the program displays the format, that indicator should be off. The program should then turn on the indicator for the next and succeeding times it issues the display. Each time the display is issued with the indicator on, the input fields are blanked out, and the operator can again use them for input. This technique is especially important when a program communicates with a remote display station because the amount of information transmitted to a remote display station might significantly affect the performance of the jobs using the communications line.

Override Fields Operation

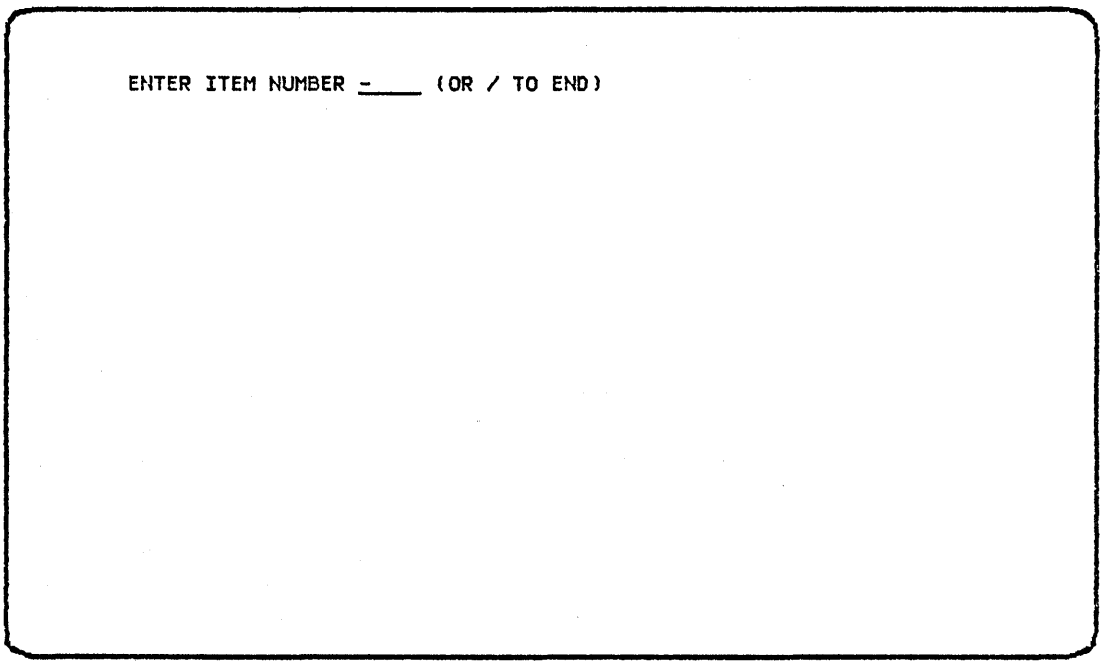
For an override fields operation, which is specified in columns 33 and 34 of the S specification, work station data management:

- Transmits the contents of conditional output fields if the output indicator is on. A conditional output field is a field for which an indicator is specified in columns 23 and 24 of the D specification. The field is transmitted only if that indicator is on.

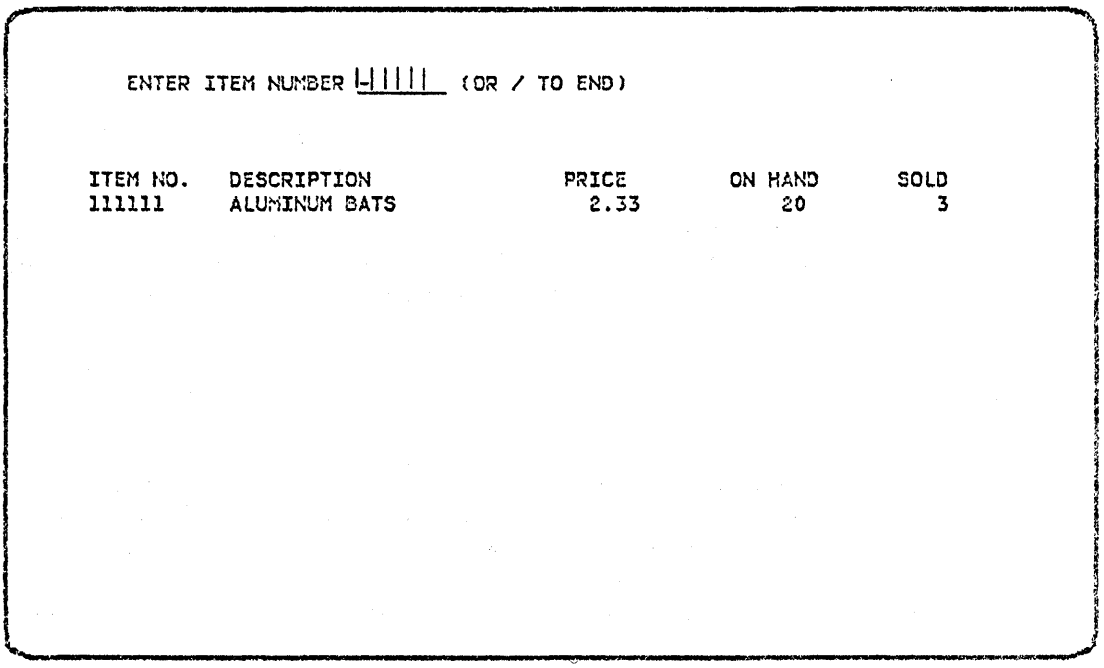
- Retransmits the attribute bytes for all field attributes controlled by indicators, except the protect field attribute.

The override operation allows the program to override (modify) program output fields on a display without retransmitting the entire display. Again, this technique is especially important when a program communicates with a remote display station, in that it reduces the amount of information transmitted over the communications line.

An example of a program that uses the override fields operation would be the following inquiry program. The first display issued by the program requests the operator to enter the item number of the item to be displayed.



After the operator keys an item number and presses the Enter/Rec Adv key, the program retrieves inventory information about the item and displays it.



If the operator enters an item number that is not in the inventory file, the program turns on the override-fields indicator and redisplays the first display. (The override-fields indicator is the indicator specified in columns 33 and 34 of the S specification for the first display.) The first display contains a message field in line 24 that is displayed only when the override-fields indicator is on. Also, the override-fields indicator could be used to redisplay the incorrect item number as a reverse-image field.

```
      ENTER ITEM NUMBER 121212  (OR / TO END)


  ITEM NO.   DESCRIPTION              PRICE       ON HAND      SOLD
  111111     ALUMINUM BATS             2.33            20         3









ITEM NOT FOUND
```

Unless the suppress input operation is requested when the override fields operation is performed, the operator can enter another item number.

Suppress Input Operation

For a suppress input operation, which is specified in columns 35 and 36 of the S specification, work station data management will *not* invite input from the display station after transmitting the format to the display station. The operator may enter information into input fields on the display, but he cannot enter the display and return the input to the program until work station data management invites input from the display station. The suppress input operation should be used when multiple formats are displayed before input is sent to the program. When multiple formats are sent, the suppress input operation should be specified on all but the last display or the system will have to do extra work to generate the displays.

Note: If an RPG II MRT program transmits multiple displays and does not suppress input on all but the last display and if the operator interrupts the program by pressing the Attn key before the last display is transmitted, *the program will be suspended.* No other requestors will be serviced during the inquiry request. No indication that the program is suspended is given to the operators at the other display stations.

Operations Requested by Basic Assembler Programs

Appendix A describes the operations that can be requested by a basic assembler program when it calls work station data management.

File Concepts

This section describes these System/34 file concepts:

- File identification
- File organization
- File processing
- File sharing in multiple program mode
- File sharing consideration
- Sector protection
- File update programs
- Key sorting for indexed files
- IPL file rebuild function
- Using a disk file as two or more logical files
- Use of a file by an inquiry program in single program mode

FILE IDENTIFICATION

The SSP requires that you must be able to uniquely identify each file on the disk. You can accomplish this by assigning a unique label, eight character maximum, to each file on the disk. However, you can assign the same label to more than one file as long as each of those files has a different creation date.

Groups of files, as well as individual files, can be uniquely identified. File labels of files that belong to a file group contain one or more periods. The characters preceding a period identify the file group. Examples of labels of files within a file group are:

A.B.GO A.B.INV A.INV A.ACCTS A.PROLL	Files in file group A
A1.INV A1.ACCTS A1.PROLL	Files in file group A1
A.B.GO A.B.INV	Files in file group A.B

Only the SAVE procedure and the $COPY and $DELET utility programs can process file groups.

If a file label does not contain one or more periods, the file is not a member of a file group. User libraries cannot be processed as file groups.

Depending upon the disk storage capacity of your system, you can create up to 2008 files and libraries on your system.

FILE ORGANIZATION

A file can have one of three types of file organization based on the arrangement of records within the file: sequential, indexed, or direct. All three types can be delete-capable files. For further information, refer to *Delete-Capable Files*, later in this section.

In a *sequential* file, the position of a record depends upon the order in which records are placed in the file. The first record placed in the file occupies the first record position in the file. Subsequent records are placed in the file sequentially.

A sequential file organization usually requires less disk storage than do indexed and direct organizations.

In an *indexed* file, an entry for each record is stored in a separate part of the file called an index. The index has a record key and record location for every record in the file. An index allows a program to process required records by referring to the record keys.

The index is divided into a primary area and an overflow area. The entries in the primary area are in ascending order by record key. The entries in the overflow area are the result of added records and are not necessarily in sequence except for IFILES. When the SSP does a key sort for the file, the SSP merges the overflow entries into the primary area. Refer to *Key Sorting for Indexed Files* later in this section for information about when the SSP does a key sort.

You can bypass the checking of duplicate keys when you are randomly adding records to an indexed file by specifying BYPASS-YES on the // FILE OCL statement. The bypass function is supported by the program-defined access path and is not a file attribute. For example, if you have a shared file, one access path used by one program may bypass duplicate key checking, while the other access path used by another program may check for duplicate keys being added to the file.

Because the BYPASS-YES parameter of the // FILE OCL statement can result in an indexed file containing records with duplicate keys, you should have a method of checking to see that records with duplicate keys are not present in an indexed file. For example, you might set an L1 indicator on to check for records with duplicate keys when using RPG II.

In a *direct* file, a relationship exists between records and their positions in the file. In a direct file, the relative position of a record might be equal to a program counter or a field value. The relative position also might be derived by a formula or conversion technique (known as an algorithm) that the programmer codes. Such an algorithm might result in a number of records producing the same relative record position. Those records are called synonyms, and a programmer must allow for their storage and retrieval in alternative locations within the file. If a direct file is created as delete-capable, unused record locations are initialized to hexadecimal FFs. If a direct file is not created as delete-capable, unused record locations are initialized to hexadecimal 40s (blanks).

Additionally, an algorithm might result in some record positions being unoccupied. The algorithm chosen should be the best compromise between wasted disk space and an unnecessarily large number of synonyms.

FILE PROCESSING

Programs process files by four basic methods: consecutive, sequential by key, random by key, and random by relative record number. These program processing methods should not be confused with physical file organizations. The following table illustrates which processing methods can be used for each file organization.

	File Organization		
Processing Method:	**Sequential**	**Indexed**	**Direct**
Consecutive	Yes	Yes[1]	Yes
Sequential by Key	No	Yes	No
Random by Key	No	Yes	No
Random by Relative Record Number	Yes	Yes[2]	Yes

[1]Only consecutive input processing is valid.
[2]Changing the key in a record when processing an indexed file randomly by relative record number results in the key in the data record being changed and not the index.

The consecutive method processes records in the order in which they physically appear in a file. This method can be used for all three file organizations. Records are read until the end of the file is reached or until the program stops reading records. If a delete-capable file is processed consecutively, any deleted records are bypassed when they are read.

Typically, sequential files are processed consecutively when all records in the file are to be processed.

When a direct file is processed consecutively, the contents of spaces left for missing records are read as blank records. If a delete-capable direct file is processed consecutively, the deleted records are bypassed when they are read. When an indexed file is processed consecutively, the index is ignored.

Sequential-by-key processing applies only to indexed files. When an indexed file is processed this way, records with entries in the primary index area are processed in the order of their record keys. Records with entries in the overflow area are also processed when an indexed file having the IFILE characteristic is processed this way. In a delete-capable file, the deleted records are bypassed when they are read.

An indexed file can be processed sequentially in one of two ways: total file by key or records within limits. Total file by key processes records in the order of their key fields. Processing continues until all records with entries in the primary area have been read or until the program stops reading records. If IFILE support is used, all records can be accessed. Processing records within limits allows a section of a file, a group of records, to be processed in key sequence. Each section is identified by a lower limit, a starting key, and an upper limit, an ending key.

Random processing by key allows records in an indexed file to be processed in a sequence determined by the user. The user program specifies the record key of the desired record. Disk data management scans the index portion of the file until the key is found and then reads the desired record directly into the user program. This access method allows the program to retrieve records from the primary area and the overflow area. When a delete-capable file is processed randomly by key, the deleted records are not available.

Random processing by relative record number allows disk records to be processed in a sequence determined by the user. (The file is treated as a direct file even though it is an indexed or consecutive file.) A particular record can be processed independently of its relation to other records. All three file organizations can be processed randomly by relative record number. Relative record numbers are used to identify records. These numbers indicate the positions of records within the file in relation to the beginning of the file. The relative record numbers are not disk addresses, but positive, whole numbers that the SSP converts to disk addresses of the records. When a delete-capable file is processed randomly by relative record number, the deleted records are not available.

If you want to process an indexed file both by key and by relative record number, you can define two logical files in your program for the same physical file. This process is described in *Using a Disk File as Two or More Logical Files* later in this chapter.

The relative record numbers can be in an ADDROUT file created by the Sort program of the Utilities Program Product. An ADDROUT file is a record address file that contains binary, 3-byte, relative record numbers that indicate the relative position of records in the file to be processed. A record can be read from the ADDROUT file and used to access a record in the disk file.

Typically, sequential files are processed randomly when only a few of the records in the file are to be processed and the user knows their relative location. To process a sequential file randomly, the program should define the file as direct.

DELETE-CAPABLE FILES

The extended disk data management function of the System/34 SSP allows you to create a delete-capable file. User programs can then delete records from such a file. Any file, regardless of organization and processing method, can be created as delete-capable. When a direct delete-capable file is created, all bytes in the file are set to hexadecimal FFs. When a record is deleted from a delete-capable file, the SSP fills the record with hexadecimal FFs. Also, for an indexed file, the key is marked as deleted, not the record.

If you use an ADDROUT sort to access records in an indexed file and you delete a record, the record is deleted, but not the key. You must reorganize the file with a SAVE and RESTORE operation to delete the key.

When a primary, secondary, or demand file containing deleted records is processed consecutively or sequentially, each deleted record (a record containing hexadecimal FFs) is bypassed and the next record is read. When a file containing deleted records is processed randomly with a CHAIN operation by RPG II the *no-record-found* indicator is turned on when a deleted record is accessed.

Note: The SSP checks the first byte of a record. If that byte contains a hexadecimal FF, the record is bypassed.

Extended disk data management will not allow a record having hexadecimal FF as its first byte to be written to a file during an output, add, or update operation. If the first byte of the record contains a hexadecimal FF, an *invalid update/add/output* completion code is returned to the user program.

You can add records to delete-capable sequential and direct files by using relative record numbers. By using relative record numbers you must add a record in the file in the place you want, if the record has been deleted before. If you are using RPG II, the relative record number of the record to be added to the file must be placed in the RECNO (continuation line of the F specification) field. The relative record number must be the record number of a deleted record. You code output specifications that contain ADD to add records to a file. RPG II uses the relative record number from the RECNO field to locate where the record is to be added to the file. If the relative record number is not the number of a deleted record, a halt occurs and the system issues a message that a duplicate record exists in the file.

If you are using COBOL and specify relative organization for the file, you can add records to delete-capable files also. When ACCESS IS RANDOM or ACCESS IS DYNAMIC is specified, new records are inserted into the file. The RELATIVE KEY specified for the file must contain the desired relative record number for this record before a WRITE is issued. When the WRITE statement is executed, the record is placed at the specified relative record number position in the file. If the relative record number is not the number of a deleted record, a halt occurs and a file status return code indicating that a duplicate record exists is returned to the program.

For RPG II programs, the delete-capable file must be defined as an update file (U in column 15 of the file description specification) if records will be deleted. To delete a specific record, DEL must be specified in columns 16 through 18 of the main output record line. The DEL applies to all OR extensions of the main line.

For COBOL programs, the DELETE statement is used to delete records.

For BASIC programs, the DELETE statement is used to delete records.

When records are loaded or added to a delete-capable direct file, the specified relative record number must be the relative record number of a record that contains hexadecimal FFs; otherwise, an error message is displayed and the record will not be added to the file.

RPG direct files that are not delete-capable must be loaded using the CHAIN operation.

DFU cannot be used to create delete-capable files or to delete records in the manner described for the system delete function. DFU can, however, be used to update, list, and inquire into delete-capable files in a manner similar to that used for nondelete-capable files.

WSU does not allow records to be deleted. Therefore, a WSU program ends abnormally if it tries to use a delete-capable transaction file. A WSU program can, however, use a delete-capable master file.

EXTENDABLE DISK FILES

Specifying extendable disk files prevents your program from abnormally terminating when there is no room in the data file to add additional records.

When you use a disk file on System/34, you can use the EXTEND parameter on the FILE OCL statement to identify the file as being extendable. For an extendable file, the SSP automatically attempts to allocate more space for the file each time it becomes full. The value of the EXTEND parameter specifies the amount of additional space that is allocated. (If the file size was originally allocated in blocks, the value of the EXTEND parameter is in blocks; if the file size was originally allocated in records, the value of the EXTEND parameter is in records.) The specified value must be at least large enough to hold one record. If a file is being shared and the various users specify different EXTEND values, the EXTEND value for the user that caused the file to become full is used for the extension. When a file is extended, all users can take advantage of the additional file space, regardless of whether or not they specified EXTEND on the FILE OCL statement.

The SSP attempts to extend an extendable file whenever it becomes full as the result of one of the following situations:

- When an indexed or consecutive file is being created or added to.
- When a direct file is being created.
- When a record is read from a direct file for updating or is written to a delete-capable file, and the specified relative record number is beyond the end of the file. (If the relative record number is greater than the file size plus the extend value, the SSP does not attempt to extend the file. In that case, a *no record found* completion code is returned to the user program.)

The following list describes how the SSP handles different types of extendable files when they become full:

- Scratch and job files in the reserve area: The SSP displays a message when such a file becomes full. The operator can then choose to have the SSP copy the file into a larger space outside of the reserve area. The user program will resume processing after the extend operation. If the operator does not choose to extend the file, he can choose an option that returns an *end of extent* completion code to the user program or an option that cancels the user program.

- Direct and consecutive files: The SSP attempts to allocate the additional space immediately following the file. If that is not possible, the SSP copies the file to a larger area on disk and frees the area originally occupied by the file.

- Indexed files: The SSP copies the file to a larger area on disk and then frees the area originally occupied by the file.

If the SSP successfully extends the file, an informational message identifying the file as having been extended is logged in the history file. Execution of the user program then resumes. If the extend operation was not successful, either because space was not available or because a disk I/O error occurred, an *end of extent* completion code is returned to the user program.

INDEXED FILES WITH THE IFILE ATTRIBUTES

IFILE support allows shared indexed sequential processing of all records in an indexed file by keeping the entries in the index overflow area in sequence. Without IFILE support, indexed sequential processing is limited only to those having index entries in the primary portion of the index. IFILE support requires that both the extended disk data management and extended index data management be selected during system configuration. Indexed files can be given the IFILE attribute by specifying it on the FILE OCL statement or the SETFILE or BLDFILE procedure.

The functions provided by IFILE support require additional system resources. These resources include processing and I/O cycles as well as additional main storage. Therefore even though the function requires no recompilation of programs, you should be selective of which files you give the IFILE attribute to. Generally, IFILE support is advantageous only in file sharing situations where indexed sequential processing is used while indexed random adds are being performed. Large files with a low volume of records being added to the file are the best candidates for IFILE processing.

The following list summarizes the general functions and restrictions of the IFILE support:

- IFILE processing allows shared indexed sequential retrieval of records that have been added by shared indexed random users.
- IFILE processing does *not* change the indexed sequential add restrictions; that is, you still cannot specify DISP-SHR with indexed sequential add operations.
- The indexed sequential sequence of get, add, and update data management operations are not supported with IFILEs.
- Any binary key of hexadecimal zeros is not supported. There is no restriction on decimal key values.
- Specifying DISP-SHR will assure that doubly defined files in one program will have access to records added by that program.
- IFILE processing does not apply to the assembler language access method called ISRI (indexed sequential random input).

Files having the IFILE attribute place additional demands upon system resources. The following list summarizes some techniques you can use to help performance on your system when using files with the IFILE attribute:

- Add records to the indexed file with the key values in ascending order if possible.
- Use the KEYSORT procedure at either IPL or STOP system time, since fewer entries in the index overflow area require less search time and improve performance.
- Allocate disk space slightly larger than the expected file capacity.

Indexed files without the IFILE characteristic need to have the added keys in the overflow area keysorted into the primary index area before records added to the file can be accessed sequentially by key. Refer to *Key Sorting for Indexed Files* later in this chapter for more information on when the SSP does a key sort.

PROGRAMMING ATTRIBUTES OF IFILES

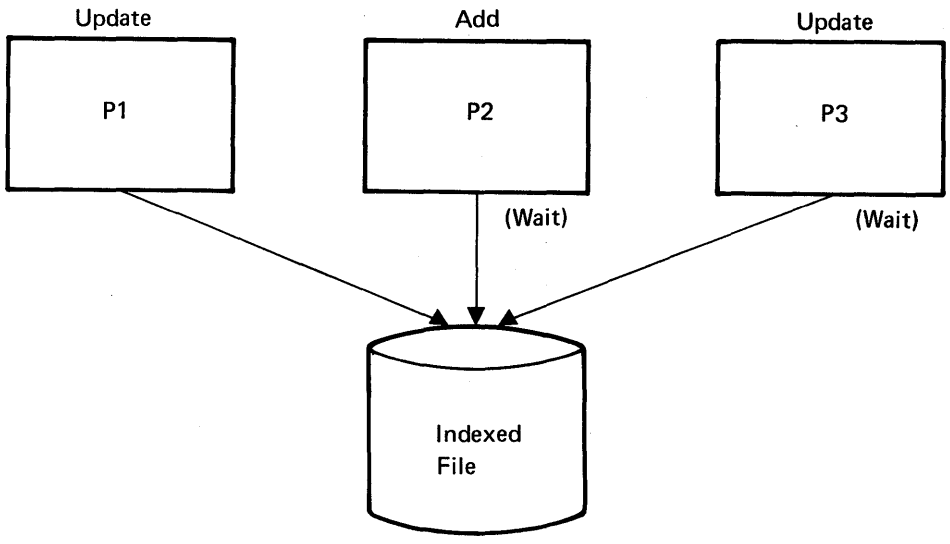
File Locking and IFILES

Beyond the normal sector protection of records being updated, IFILEs have an additional level of file protection: the entire file is locked so that the index entries can be maintained in sequence. This additional level of protection can cause some problems when a program updating records and a program adding records are sharing the same indexed file. Record update programs mark the file as being in use until the record update is complete and another operation is issued by the same program. The record-adding program begins when the operation is issued.

This condition occurs when any of the following happen:

- Specifying IFILES as extend capable files.
- Accessing data records with index entries in the overflow portion of the file.
- Accessing records in the same sector as the next record slot.
- The area in the file where the next new record is to be placed is the next record slot.

For example, assume three programs, P1, P2, and P3, are sharing an indexed file that is an IFILE. Program P1 uses the file first to update records. Program P2 is adding records to the file, and program P3 is updating records. Programs P2 and P3 wait until the record updating done by program P1 is complete and program P1 does another operation.



Performance Considerations

Performance with IFILEs can be slowed down by:

- A large overflow area in the indexed file
- A program that occupies a large area of main storage
- A small user area in main storage

These three factors affect the way the system uses I/O buffers for IFILE support.

Performance may also slow down when adding records randomly by key to an indexed file that has not had a keysort done for some time.

Keysorts and IFILES

Specifying an indexed file as an IFILE eliminates the need for doing a keysort each time you want to sequentially process the indexed file to gain access to new records added to the file.

However, System/34 automatically keysorts IFILEs when:

- A job step terminates normally and the file is not a shared indexed file
- The last program using the shared index file ends

FILE SHARING IN MULTIPLE PROGRAM MODE

System/34 supports file sharing in multiple program mode; file sharing allows two or more programs to access the same file. DISP-SHR must be specified on every FILE OCL statement for a file that is shared. For BASIC programs, SHR must also be specified on the OPEN statement for the file.

Types of Files That Can Be Shared

Input and update files can always be shared. Add files, except for indexed add files that are processed sequentially by key, can always be shared.

Types of Files That Cannot Be Shared

Files that are being created cannot be shared. An indexed add file that is processed sequentially by key cannot be shared.

Accessing Records Added to Shared Files

Programs sharing sequential files always have access to added records. Programs sharing an indexed file have access to added records, except in either of the following situations:

- If records have been added to a file that is being processed sequentially by key, the program cannot access records added to the file since the last time the keys were sorted unless the file is an IFILE.

- If records have been added to a file that is being processed consecutively, the program cannot access records added to the file since the file was allocated to the program.

Refer to *Key Sorting for Indexed Files* later in this section for information about when the SSP does a key sort.

If a program that is adding records to a shared file is ended abnormally, the records added before the abnormal termination remain in the file. If a program that is adding records to a file that is not shared is ended abnormally, records added before the abnormal termination are removed from the file unless the file is an indexed file using the IFILE support. If the system unexpectedly stops (for example, as the result of a power failure), added records can be recovered during the IPL file rebuild process. For information about IPL file rebuild, refer to *IPL File Rebuild Function* later in this section.

FILE SHARING CONSIDERATIONS

If you want to share a file among programs or if you want to allow more than one display station to process a file concurrently, consider that:

- Programs can share only permanent or temporary files.
- If you change a temporary file to a scratch file by specifying a RETAIN-S parameter on the FILE OCL statement, the file cannot be shared.
- The SSP protects sectors read for possible updating by one program from being updated by any other programs. The SSP protects sectors by assigning them to the program doing the update. Refer to *Sector Protection* in this section for further information.
- If programs share more than one file, all programs should access the files in the same sequence to reduce the chances of a deadlock condition. Refer to *Sector Protection* in this section for a description of this deadlock condition.

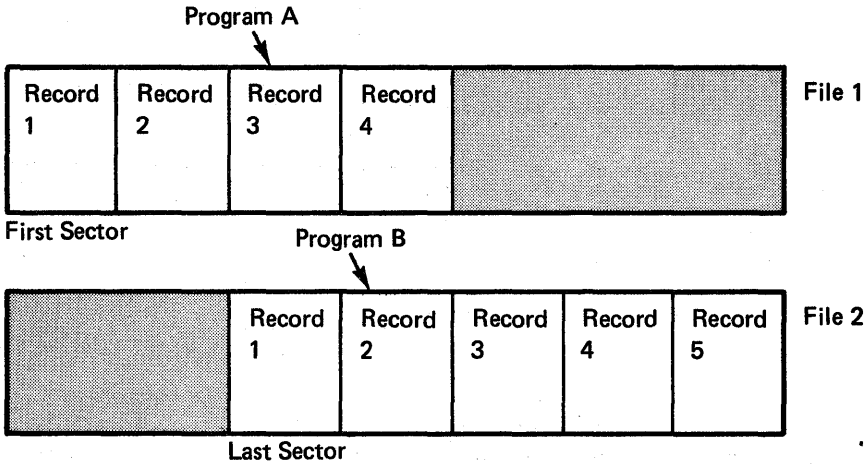
SECTOR PROTECTION

Sector protection is the SSP function of assigning sectors to a program when that program reads records in the sectors for a possible update. A sector is a 256-byte area on disk and is the smallest amount of data that can be read from or written to disk during one operation.

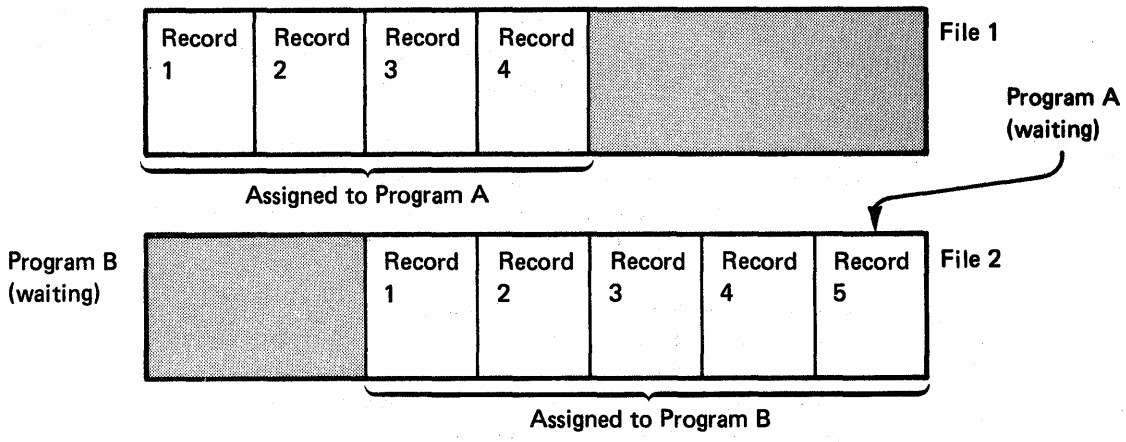
How you set up record blocking within your application program affects the number of records that are sector protected. A sector can contain more than one record, and more than one sector can be assigned to the input blocks you specify in your application program. Each record contained within a sector is sector protected. For more information on how input blocks and sectors are related, refer to *Record Blocking* in the next chapter.

A program that tries to update a record within a sector that is assigned to another program must wait until the sector is released. A program releases a sector by (1) completing the update, (2) performing an add operation, (3) reading a different sector from the same logical file or (4) rereading the same sector. (For information about logical files and physical files, refer to *Using a Disk File as Two or More Logical Files*, later in this chapter.) Programs that request an assigned sector for input only do not have to wait until the sector is released.

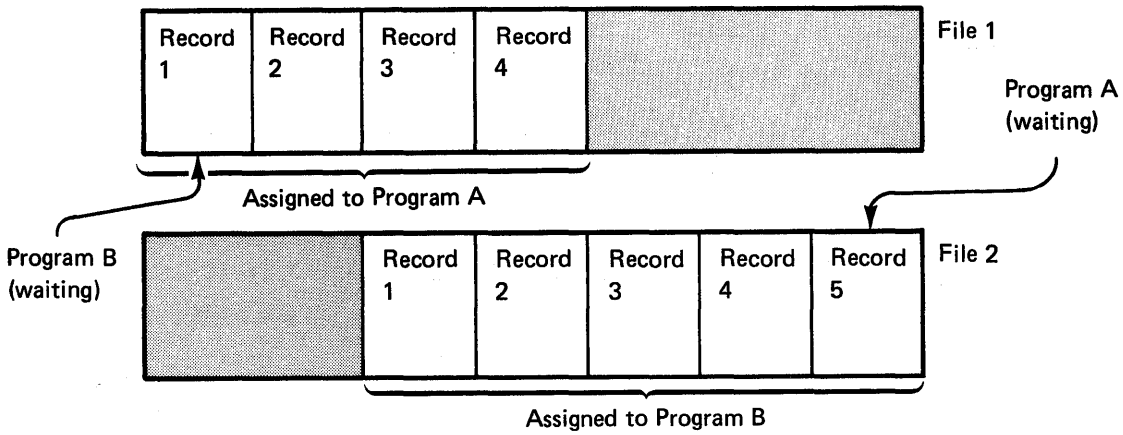
A deadlock condition can occur when update files are shared. For example, assume that program A and program B are updating two shared files, file 1 and file 2. Program A reads record 3 updating from the sector of file 1, and program B reads record 2 for updating from the last sector of file 2.



Suppose that program A tries to read record 5 from the last sector of file 2. Program A must wait because the sector is assigned to program B.



Suppose that program B tries to read record 1 from the first sector of file 1. Program B must wait because the sector is assigned to program A.



This condition of programs waiting for each other is called a deadlock. To ensure that deadlocks do not occur and that a sector from a shared update file is not assigned for a long time, always release a sector before reading a record from another shared update file. A sector is released when it is rewritten to disk, or a different sector is read.

For example, in RPG II programs, a good way of releasing a sector is to write an output record that has no fields (columns 32 through 71 of the RPG II output specifications specify no data). This rewriting causes the sector to be released and requires no physical disk accesses.

Note: If the operator selects inquiry option 1 while the interrupted program has a sector enqueued, the system does not resume execution of the interrupted program until all sectors are released. The system then allows the operator to enter another command or statement.

Interrupting a program that is updating records in an IFILE can cause a file lockout condition which prevents you from accessing records in the file. The file lockout condition can occur when a program updating records in the IFILE is interrupted. For example, an operator presses the ATTN key and takes option 1. The operator starts a program to add records to the file already opened by the program that was interrupted.

FILE UPDATE PROGRAMS

You should use care when updating disk files in any program that supports multiple display stations. If a single logical file is used by two or more display stations within the same program and if the program, after reading a record for updating, does other read operations from the same file before actually updating the record, the following error conditions can occur:

- An update or part of an update can be lost. For example, suppose a record is read from File X and displayed at display station 1; then the same record is read from File X and displayed at display station 2. If File X is not shared, the update done by one display station might be destroyed by an update done by the other display station. If File X is shared, an error message is issued, and the second update is not done.

- The wrong record can be updated. For example, suppose a record is read from File X and displayed at display station 1; then a different record is read from File X and displayed at display station 2. If display station 1 tries to update the first record but does not reread that record, disk data management tries to update the last record read from File X.

- An update performed by another program sharing the file can be lost. For example, suppose a record is read from File X and is displayed at display station 1; then a record in a different sector of File X is read by the same program and displayed at display station 2. The second read from File X causes the SSP to free the sector containing the first record. Another program that is sharing File X can update the first record. If display station 1 rereads and also tries to update that record using the original field values, the updates made by the other program might be lost.

You can avoid the preceding error conditions by using one of the following techniques:

- Before doing an update, reread the record and check that none of the fields being updated have been changed since the record was displayed for updating. If any of the fields were changed, redisplay the field for updating or, if possible, do the update using the field values currently in the record.

- Protect records being updated by establishing a field in the disk records to be used as a busy indicator. After reading and displaying the record for update, place the busy indicator in the record and write the record to disk. (The busy indicator might be the work station ID and the program name.) Subsequent attempts to access the same record should test for the busy indicator and not allow access for update. The busy indicator should be removed from the record when the update is done by the requestor or if no update is to be done. If the possibility exists that another program might update the same file concurrently, both programs must test and use the same busy indicator.

  If the program ends abnormally and you are not going to restart the program, you should run another program that turns off the busy indicators in records that were being updated by the program when it ended so that programs that check the busy indicator can handle the record properly.

- Consider defining a separate logical file for each display station. Separate logical files protect against updates by other programs, but they do not protect against multiple updates within a single program. For further information, refer to *Using a Disk File as Two or More Logical Files*, later in this section.

KEY SORTING FOR INDEXED FILES

After an indexed file is created or after an indexed file has records added to it, the added keys exist in an overflow area. If the file is to be processed sequentially by key, the keys must be sorted to allow access to all records in the file, unless IFILE support is being used. Even when IFILE support is being used, key sorting is done by the system to limit the size of the overflow area.

The SSP sorts keys at the end of a job if the indexed file was created during the job and the records were not placed in the file in key sequence. Additionally, IFILEs are keysorted at end of job when the system detects that add operations are slow because of the size of the overflow area.

The SSP performs a special key sort at the end of a job when certain conditions are met. The special end-of-job key sort is done if all of the following conditions are true:

- The file is being used exclusively by one user, or the file is being used by the last shared user.
- There was no overflow key area for this file when the file was first allocated.
- All keys added to the file are in ascending sequence, and the value of the added keys is greater than the highest key in the primary key area of the file.

The SSP sorts keys when a file is allocated to a program if *all* the following conditions are true:

- The file has keys out of sequence because records were added to the file.
- The file processing method to be used is sequential by key.
- The file is not shared.
- The file is not being used by an interrupted program.

The system operator can request that the SSP sort keys at system shutdown. To request a key sort, the operator can specify STOP SYSTEM or STOP SYSTEM,SORT. The SSP sorts keys of files that (1) have had records added to them and (2) have not had the keys sorted. That is, keys are sorted for files that have overflow areas.

The SSP sorts keys during an IPL if the system operator enters Y in response to the EXAMINE AND VERIFY THE DISK VTOC? (Y,N) prompt on the file rebuild display and if the sort or merge flag is set on in the VTOC entry. These flags are set during file processing whenever a duplicate key or an out-of-sequence key is found. The flags may be set by disk data management, diskette-to-disk copy, or index construction during IPL file rebuild. Files might have keys out of sequence if a system shutdown was not done before the IPL.

The SSP sorts the keys of files processed by some SSP utility programs. When the following SSP utility programs or procedures are used to do the indicated functions, the SSP sorts the keys in the file:

SSP Procedure	SSP Utility	Function
RESTORE	$COPY	Restore a file from diskette
DISPLAY	$COPY	Display an indexed file
ORGANIZE	$COPY	Organize a file to disk or diskette
TRANSFER	$BICR	Transfer a file from disk to diskette
KEYSORT	$DDST	Sort the keys of an indexed disk file

The SSP does not sort the keys of shared files. Therefore, records added to those files cannot be accessed when the file is processed sequentially by key, unless IFILE support is being used. To be able to access these added records, you can use the KEYSORT procedure or the $DDST utility program. The KEYSORT procedure does not execute until it has exclusive use of the file.

The KEYSORT procedure issues a warning message when records with duplicate keys are found in an indexed file.

IPL FILE REBUILD FUNCTION

System/34 provides IPL file rebuild as a recovery function that (1) verifies the integrity of the VTOC entry for each user data file and (2), if possible, corrects the entry and/or the file itself. Therefore, the IPL file rebuild function should be run during IPL following failures such as power outages and inadvertent IPLs.

The IPL file rebuild function is controlled by options on the second IPL display. From that display, the operator can (1) request file rebuild or bypass it, (2) limit the function to only files that were being created or to all files, (3) request that the labels of all files in error be displayed, (4) request that the VTOC entries for nonrecoverable errors be deleted, and (5) request that checkpoint/restart active files be deleted.

If disk file reorganization (COMPRESS) had not completed when the system failure occurred, the IPL file rebuild function calls the $FREE utility to complete the reorganization.

The IPL file rebuild function then examines each entry in the VTOC. If a VTOC entry contains any of the following errors, the entry cannot be corrected:

- The file type is not sequential, direct or indexed
- The retention type is neither temporary nor permanent
- The record length exceeds 4096 bytes
- The file is an indexed file, and the key length is zero or exceeds 29 bytes
- The file is an indexed file, and the key position exceeds 999 bytes
- The file is an indexed file, and the key is not within the record
- The file is not located within the data file bounds

Because System/34 initializes the data space to hexadecimal zeros for new sequential and indexed files, to blanks for direct files, and to hexadecimal FF for delete-capable direct files, the IPL rebuild function can update the end-of-data pointer in the VTOC entry to reflect records that were added to the file.

Note: Records added to a file may be written on disk or may be in a main storage buffer. If a failure occurs, only those records written on disk can be located in order to update the VTOC entry.

If the file is an indexed file, the entire index is reconstructed if (1) the end-of-data pointer is updated, (2) the sort or merge flag is set on in the VTOC entry, or (3) the number of indexes is not equal to the number of records.

In the VTOC entry for an indexed file, a flag (sort or merge) is set on during file processing whenever a duplicate key or an out-of-sequence key is encountered. This flag may be set on by disk data management, diskette-to-disk copy, or index construction during IPL file rebuild. If the flag is set on, the IPL file rebuild function calls the key sort function.

Finally, the IPL file rebuild function ensures that all files with unique labels are marked with the latest date indicator in the VTOC entry. For files with the same label, the dates of creation must be unique, and only the file with the most recent date is so marked.

The IPL file rebuild function then re-creates the disk control block to reflect both available space and space in use on the disk.

USING A DISK FILE AS TWO OR MORE LOGICAL FILES

Each file defined within a program is called a logical file. A program can use one disk file as two or more logical files. An RPG II program, for example, may be written to access two files called FILEA and FILEB. If the following FILE OCL statements are used:

```
// FILE NAME-FILEA,LABEL-MASTER,DISP-SHR
// FILE NAME-FILEB,LABEL-MASTER,DISP-SHR
```

then the disk file labeled MASTER is used as two different logical files by the program.

An example of using multiple logical files could be a bill of materials program that accesses a master file randomly by key *and* randomly by relative record number.

Records added to one logical file can be accessed from a second logical file except in the following cases:

- Records in the second logical file are being read or updated, and the file is an indexed file being processed sequentially by key (unless the file is an IFILE).
- The second file is an input-only file that is being processed consecutively.

If DISP-SHR is not specified on the FILE statements, records can be added or updated in only one of the logical files. If DISP-SHR is not specified on the FILE statements for multiple logical files, the following considerations exist:

- Records can be added or updated through only one of the logical files.
- If the input-only file(s) is being processed sequentially by key or consecutively, added records cannot be read. In addition, updates to records that have already been read into the I/O buffer are not available to the input-only file unless it is an IFILE. For information about when records are read into the I/O buffer, refer to *Physical I/O and Logical I/O* in Chapter 3.
- If the input-only file(s) is being processed randomly (either by key or by relative record number), the file has immediate access to updated records.
- If the input-only file(s) is being processed randomly by relative record number, added records cannot be read.

If DISP-SHR is specified, records can be added or updated in more than one of the logical files. Care must be taken, however, because the SSP does not protect against two logical files updating or adding to the same sector at the same time. If two logical files are updating or adding to the same sector at the same time, only the update or add made last will appear in the disk file.

Note: If the file is shared, the system does protect each logical file against concurrent updates by other programs. The system does not protect against concurrent updates from within the same program.

USE OF A FILE BY AN INQUIRY PROGRAM IN SINGLE PROGRAM MODE

In single program mode, the SSP allows an inquiry program to access active files for input or update. Active files are those files that were being used by the program that was suspended by the inquiry request. Updating an active file is allowed only if the suspended program did not open the file for an update or add operation. Active output files can never be accessed by an inquiry program.

Figure 2-4 summarizes the types of file processing that can be used by an inquiry program when processing a file that was being used by the suspended program.

Interrupted Program

Inquiry Program: Access Method		File Type: Indexed IS I	IS O	IS U	IS A	IR I	IR O	IR U	IR A	C I	R I	R U	Sequential C I	C O	C U	C A	R I	R U	Direct C I	C U	R I	R O	R U
IS	I	y	n	y[2]	y[4]	y	n	y[2]	y[4]	y	y	y[2]	[5]										
	O	n	n	n	n	n	n	n	n	n	n	n											
	U	y[3]	n	n	n	y[3]	n	n	n	y[3]	y[3]	n											
	A	n	n	n	n	n	n	n	n	n	n	n											
IR	I	y	n	y[2]	y[1]	y	n	y[2]	y[1]	y	y	y[2]											
	O	n	n	n	n	n	n	n	n	n	n	n											
	U	y[3]	n	n	n	y[3]	n	n	n	y[3]	y[3]	n											
	A	n	n	n	n	n	n	n	n	n	n	n											
C	I	y	n	y[2]	y[1]	y	n	y[2]	y[1]	y	y	y[2]	y	n	y[2]	y[2]	y	y[2]	y	y[2]	y	n	y[2]
	O	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
	U												y[3]	n	n	n	y[3]	n	y[3]	n	y[3]	n	n
	A												n	n	n	n	n	n					
R	I	y	n	y[2]	y[1]	y	n	y[2]	y[1]	y	y	y[2]	y	n	y[2]	y[2]	y	y[2]	y	y[2]	y	n	y[2]
	O	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n	n
	U	y[3]	n	n	n	y[3]	n	n	n	y[3]	y[3]	n	y[3]	n	n	n	y[3]	n	y[3]	n	y[3]	n	n

[1] The inquiry program can access all records in the file except those records added by the suspended program.
[2] Records retrieved by the inquiry program may not reflect the current status of the file.
[3] Records retrieved by the interrupted program may not reflect the current status of the file.
[4] The inquiry program cannot access records in the overflow area.
[5] Shading indicates an invalid access method.

Abbreviation Used	Meaning
A	Add
C	Consecutive
I	Input
IR	Random by key
IS	Sequential by key
n	Processing by the inquiry program is not allowed
O	Output
R	Random by relative record number
U	Update
y	Processing the inquiry program is allowed

Figure 2-4. Use of a File by an Inquiry Program in Single Program Mode

Inquiry Programs and IFILES

Suspending the execution of a program that is updating records in an indexed file that has the IFILE designation and then executing a program that adds records to this same indexed file may result in a condition known as an enqueue failure. An enqueue failure is a condition where a program cannot access the file. If an enqueue failure occurs, the Input Inhibited indicator on your work station goes on until you press the ATTN key and take either the 2 or 3 option.

OFFLINE MULTIVOLUME FILES

An offline multivolume file is a sequential file that is used as though it completely resides on disk but, in fact, is stored on one or more diskettes. The portion of the offline multivolume file stored on one diskette is called a file segment. Offline multivolume file processing uses a disk file as a work area in which the offline multivolume file is processed a segment at a time.

An offline multivolume file is created when a FILE OCL statement for a diskette file has the same NAME parameter as that on a FILE OCL statement for a new disk file. For example, a program creates a disk file called PAYMSTR, which requires 800 blocks of disk storage; however, 800 blocks of disk storage are not available. In this case, you can create the file as an offline multivolume file by using the following FILE statements:

```
// FILE NAME-PAYMSTR,UNIT-F1,BLOCKS-96,RETAIN-S
// FILE NAME-PAYMSTR,UNIT-I1,RETAIN-20,
//      PACK-666666,LOCATION-M1.01,AUTO-NO
```

Note: This example uses a diskette 1 diskette that has been initialized in the 128-bytes-per-sector format. The 96-block segment was selected to fully use each diskette, but you might decide to specify smaller segments if disk space is limited.

Now, when PAYMSTR is created, the system places records in the 96-block file on disk. When all 96 blocks are full, the system copies the disk file to the diskette in location M1.01. The system places the next record written by the program into the first record position in the disk file. When the file is again filled, the system copies it to the diskette in location M1.02. If all diskettes in slot M1 are filled in this manner, the system prompts the operator to insert another magazine in slot M1 and continue processing. (If AUTO-YES has been specified, the system does not prompt the operator, but uses the magazine in slot M2.) The system copies the last file segment to the diskette when the job ends.

To process an existing offline multivolume file, you must again specify two FILE statements with the same NAME parameter. The size specified on the FILE statement for the disk file must be the same as the size specified when the file was created.

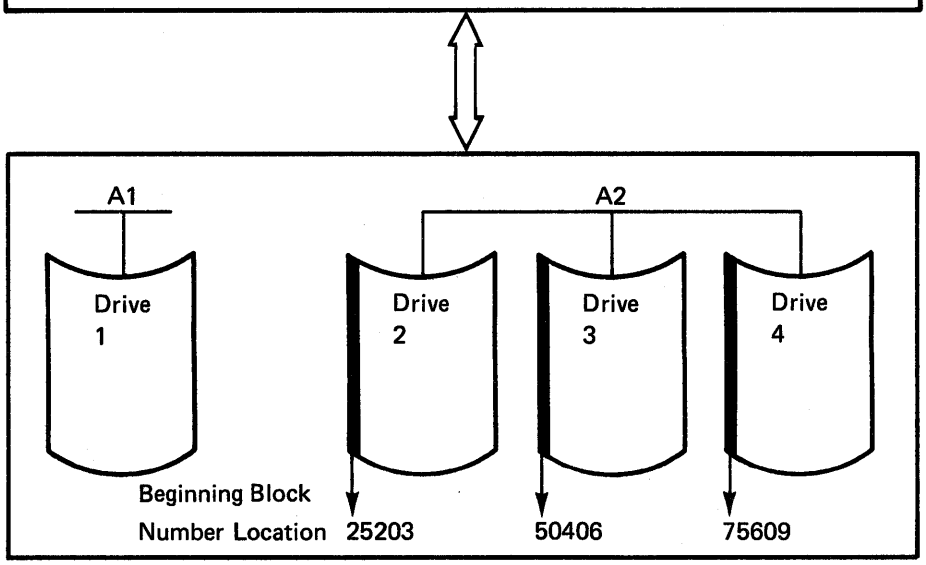
The following restrictions apply to offline multivolume files:

- They cannot be shared.
- They can only be sequential files and can only be processed by consecutive processing methods.
- They can only be processed by means of the diskette magazine drive.
- Each access to an offline multivolume file must start with the first diskette of a magazine. Therefore, M1.01 or M2.01 must be specified on the LOCATION parameter of the FILE statement for the diskette file.
- Records added to an offline multivolume file must be added at the end of the file. Therefore, the magazine containing the last segment in the file should be inserted when an addition is to be made to the file. When additions are made, the current program date becomes the creation date of all the new segments and of the last segment of the old file.
- When segments are added to an offline multivolume file, the diskettes must not contain any other active files.
- Only one offline multivolume file at a time can be processed by a step.
- A program running in inquiry mode cannot access an offline multivolume file that was being used by the interrupted program.
- Offline multivolume files created by System/32 cannot be processed by System/34. The offline multivolume files created by System/34 cannot be processed by System/32.
- The size of the disk file cannot exceed the capacity of an individual diskette:
  - For a diskette 1 diskette initialized in the 128-bytes-per-sector format, the maximum file size is 96 blocks.
  - For a diskette 1 diskette initialized in the 512-bytes-per-sector format, the maximum file size is 118 blocks.
  - For a diskette 2D diskette initialized in the 256-bytes-per-sector format, the maximum file size is 384 blocks.
  - For a diskette 2D diskette initialized in the 1,024-bytes-per-sector format, the maximum file size is 473 blocks.

THIRD AND FOURTH DISK DRIVE IMPLEMENTATION CONSIDERATIONS

You have additional disk capability and disk seek processing with the third and fourth disk drives on the System/34. The SSP supports the additional disk drives by extending the current definition of the logical name for disk drive A2. The following chart shows how the physical configuration of the disk drives and the logical disk name are related.

Physical Drive Configuration	Logical Disk Name A1	A2
1	1	–
1,2	1	2
1,2,3	1	2,3
1,2,3,4	1	2,3,4



The SSP allocates space for the file you have specified on the // FILE statement by the following rules if you did not specify a location based upon block number:

- If you specify disk A1, the file is allocated in the first segment (lowest address) on disk A1 that is large enough to contain the file. If not enough space is available on drive A1, the SSP attempts to allocate the file on the disk A2.

- If you specify disk A2, the file is allocated in the last segment (highest address) on disk A2 that is large enough to contain the file. On a three-drive system, this is the last segment of drive 3. On a four-drive system, this is the last segment of drive 4. If not enough space is available on the last physical disk of A2, the SSP attempts to find space on the other drives that comprise logical disk A2. If not enough space is available, the SSP attempts to allocate the file on disk A1.

The following chart shows the placement of files based upon file type and disk drive configuration if you do not specify a preferred disk placement by block number.

Type of File	Number of Disk Drives			
	1	2	3	4
Permanent	A1	A2	A2	A2
Temporary	A1	A2	A2	A2
Scratch	A1	A1	A1	A1
Job	A1	A1	A1	A1

Notes:

1. Permanent and temporary files are allocated in the last segment of the disk large enough to contain the file.
2. Scratch and job files are allocated in the first segment of the disk large enough to contain the file.
3. If not enough space is available for placement on a particular disk, the SSP can allocate a file that spans the disks.

The manner in which you arrange your files on the disk can affect the performance of your system. You should arrange the files on the disk so that the utilization of the disk drives is approximately the same.

For more information on how you can obtain performance data about your disk drives, refer to the *System Measurement Facility Reference Manual.*

Printer Concepts

Five types of printers are available with System/34: the IBM 5211 Line Printer, the IBM 3262 Line Printer, the IBM 5224 and 5225 Matrix Line Printers, and the IBM 5256 Serial Printer. These printers are described in the *System/34 Introduction*. Operating information for these printers is described in the following manuals:

IBM 5211 Models 1 and 2 Component Description and Operator's Guide

IBM 3262 Models A1 and B1 Component Description and Operator's Guide

IBM 5224 Printer Models 1 and 2 Operator's Guide

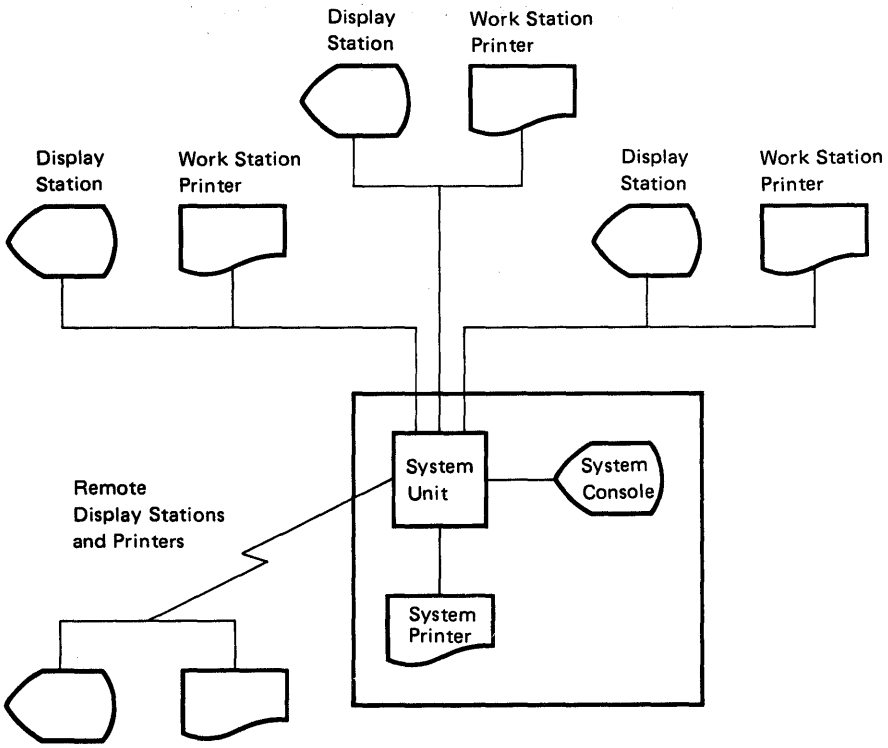
IBM 5224 Printer Models 1 and 2 Setup Procedures

IBM 5225 Printer Operator's Guide

IBM 5256 Printer Operator's Guide

One 5211 or 3262 printer can be attached to a System/34. Multiple 5256 or 5224 or 5225 printers can be attached to a System/34, either locally or by a communications line.

The system printer, assigned during system configuration, should ordinarily be located with the system unit and the system console. Any type of printer can be the system printer. All printers other than the system printer are called work station printers.



When a program prints information, the program uses either the system list function or the printer data management function of the SSP. The following diagram lists those programs that use the system list function and those programs that use the printer data management function.

Programs That Use System List (prints on the system list device for the session)	**Programs That Use Printer Data Management (defaults to the configuration printer)**
• SSP Utility Programs, except data communications and service aid utilities. Output from the menu build, history display, display screen format generator, and library maintenance procedure use system list.	• Print key • SSP data communication programs • RPG II compiler • Utilities Program Product (SEU, the job execution portion of DFU, SDA, and WSU)
• Privileged user-written assembler programs	• User-written programs • SSP Service Aids • COBOL compiler
• The job setup portion of DFU	• FORTRAN compiler • Assembler
• The sort portion of the Utilities Program Product	• BASIC

Printer Data Management Output

For individual files, you can direct printer data management output to a specific printer by using a PRINTER OCL statement. If output is not directed to a specific printer by a PRINTER OCL statement, printer data management output is printed on the default printer assigned to the display station. The default printer is assigned during system configuration and is shown as the CONFIGURATION SYSLIST DEVICE on the second session status display. You can use the SET procedure to change the default printer.

System List Output

System list output is printed on the system list device for the session. The system list device for the session is shown as the SESSION SYSLIST DEVICE on the second session status display. When an operator signs on, the default printer becomes the system list device. The SYSLIST statement or procedure can be used to change the system list device for a session.

If printed output is generated by a job run from the input job queue or from a job that was run by an EVOKE OCL statement, the system printer is used unless a PRINTER OCL statement directs output to a specific printer or unless the printer default for released jobs is changed during system configuration.

Example of Directing Printer Data Management Output and System List Output

In this example, printer P1 was assigned to a display station during system configuration. Now, however, the programmer would like to redirect all printer data management output to printer P2. The following command statement changes the default printer to P2:

```
SET ,,,,,,P2
```

After the SET statement is entered, all printer data management output is printed on P2 unless the output is directed to another printer via a PRINTER OCL statement. System list output continues to print on printer P1 or the default printer for released jobs assigned during system configuration. After the SET statement is entered, the second session status display shows P2 as the CONFIGURATION SYSLIST DEVICE and printer P1 as the SESSION SYSLIST DEVICE. If the operator then signed off and back on, both the CONFIGURATION SYSLIST DEVICE and the SESSION SYSLIST DEVICE are P2. If an IPL is performed, the default printer is not set to P1, but remains P2 (the printer specified with the SET procedure).

Vertical Line Spacing Support for the 5225 Printer

If you have a 5225 Printer, you have the ability to specify the vertical line spacing on your output reports without requiring the operator to set the hardware switch on the printer.

You can specify a vertical lines-per-inch value either during system configuration or by specifying the vertical lines-per-inch value on a FORMS or PRINTER OCL statement or executing the LINES procedure.

The default value is six lines per inch. You can specify either four, six, or eight lines per inch. If you do not specify a vertical lines-per-inch value on either the PRINTER or FORMS OCL statements or the LINES procedure, you will obtain the value specified during system configuration.

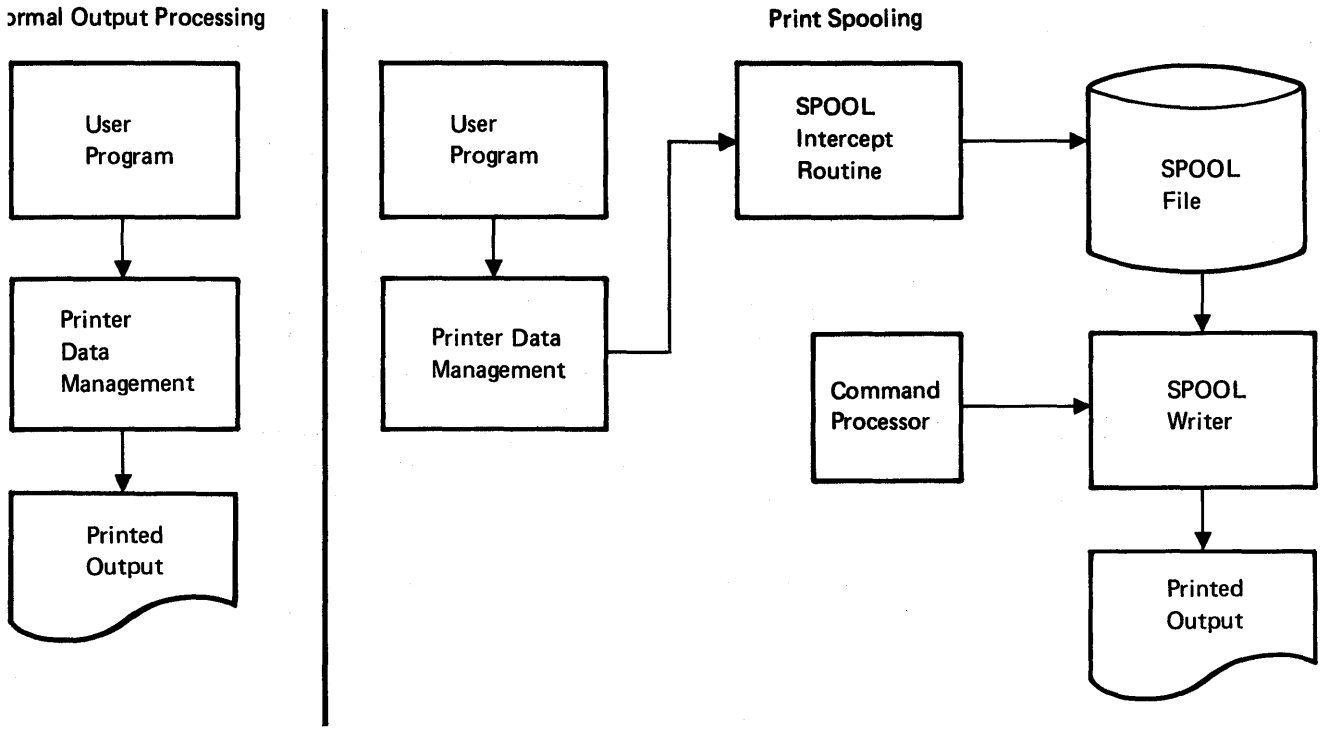
If you specify the vertical lines-per-inch values using OCL, you should specify either the FORMSNO parameter or the LINES parameter when using the PRINTER or FORMS OCL statements. This will help in setting the correct number of lines per page of forms mounted on the printer and helps in maintaining the proper page alignment on the forms when you change the vertical lines-per-inch value. You should initially run CNFIGSSP to assign the default values for vertical line spacing for your 5225 Printer. If you do not run CNFIGSSP, the SSP assumes a value of six vertical lines per inch.

If you specify a lines-per-inch value for a printer other than the 5225 Printer, the value you specify is ignored by the system. The only way you can specify a lines-per-inch value on printers other than the 5225 is by the use of a switch on the printer.

PRINT SPOOLING

Print spooling is the capability of the SSP to save printer output on disk, in an area called the spool file, for printing at a later time. The spool intercept routine saves the print data in the spool file, and the spoolwriter retrieves the print data from the spool file and prints it. The operator can control the spooling process using spool commands.

The following diagram shows the normal output process and the print spooling process.



Advantages of Print Spooling

Print spooling provides several advantages over normal printing:

- Programs execute faster because time does not have to be spent waiting for the printer to print each line of data.
- Multiple programs using the printer may be run concurrently rather than serially because they do not have to wait for the printer to become available.
- Programs producing printer output can be run even if the printer is not working.
- Multiple copies of the printer output can be produced without repeated execution of the program producing the printer output.
- Different priorties can be assigned to the printer output in order to schedule printing sequences.

- In the event of a printer malfunction, printing can be restarted without re-execution of the program that produced the printer output.

- A spool writer printing data from the spool file makes more efficient use of the main storage processor, the printer, and the communications lines (for remote printers) than direct printing does.

Spooling Options During Configuration

During system configuration the user can decide whether to include print spooling in the system. If selected, print spooling can be later cancelled via the IPL overrides or temporarily disabled for a particular printer by using the SPOOL-NO parameter on the // PRINTER OCL statement.

If print spooling is selected, you can specify whether all printers are to be spooled, or just the system printer. If only the system printer is selected, the other printer can be spooled temporarily by the use of the SPOOL-YES parameter on the // PRINTER OCL statement.

Control of Print Spooling

Print spooling may be controlled either by the system operator or by subconsole operators. Subconsoles are display stations assigned control during system configuration over one or more printers. The system operator has control over all printers and all print files in the spool file. Subconsole operators have control over their designated printer(s) as well as all print files in the spool files to be printed on their designated printers. Display station operators have some control over print files they create.

Spool File

The spool file resides on fixed disk and consists of a primary file and up to five additional areas called extents. When spooling is active, the primary file always exists; the extents exist only as needed. A new extent is allocated only when the primary file and all currently allocated extents are full. When an extent becomes empty, it is deleted.

Spool File Size

The size of the primary spool file is specified in blocks during system configuration and by the IPL overrides. When there is sufficient disk space, the amount specified is used to allocate the primary file and extents. Whenever there is insufficient disk space, the largest space available is used to allocate the primary file first.

The primary file and extents are divided into contiguous areas called spool file segments. The size of the segments is specified in blocks during system configuration and by the IPL overrides. The first segment of the primary file called the spool master segment keeps track of the rest of the spool file. All other segments of the primary file and all segments of the extents store print data. When a printer file begins producing output, a spool file segment is allocated to that printer file to store print data in. If there is more print data than the segment can contain, another segment is allocated when the first one is filled. This process continues until all the print data has been put in the spool file. Any unused space at the end of the final segment is ignored. Once the print file has been printed, it is removed from the spool file, and the segments it used become available for use by another printer file.

Refer to the *Installation and Modification Reference Manual* to determine the spool file and segment sizes based on the expected spool file usage. The following explains some of the considerations that can be made:

- The recommended size for the primary spool file is one-sixth of the number of blocks needed to contain all the print data, so that it may be spread across the primary file and all five extents. This is done so that the spool file occupies less disk space when there is less data in it. A larger primary file involves more work by the system to allocate and deallocate spool file extents. However, the spool file will occupy more disk space even when it contains less data.

- The recommended size for the spool file segments is based on the size of the typical printer file. Larger segments reduce the work the system does allocating segments because fewer segments are needed. However, smaller segments make more efficient use of spool file space by reducing the amount of unused space in the last segment of the print files. The primary file and extents have a limit of 800 segments, so in the cases where a large spool file is needed, you may need to increase the segment size to keep the number of segments from exceeding the limit.

Heavy usage of the spool file by spool routines and other programs running on the system can create increased demand for the disk. To help balance the activity across the disks, you can specify a disk preference for the spool file, both during system configuration and by the IPL overrides.

Spool Intercept Routine

The spool intercept routine stores printer output in the spool file. Printer output can be from multiple printer files, either from multiple programs producing printer output or from multiple printer files within one program, or both.

Spool intercept routine handles all printer files concurrently but independently of one another. Each printer file has its own intercept buffer to receive printer data, its own spool file segment for storing the data in the spool file, a unique spool ID consisting of the characters SP followed by four decimal digits.

Each printer file may be opened, closed, and reopened, etc. as desired independently of all other printer files. When a printer file is closed, it becomes disassociated from the spool intercept routine. If the printer file is reopened, the spool intercept routine treats it as an entirely new printer file. There is no limit to the number of printer files from one or more programs that can be handled concurrently by the spool intercept routine.

If the spool file is not large enough to contain all the print data being put into it, a message is issued when it becomes full. You may have to wait until space becomes available in the spool file and then respond to the message with an option that allows processing to continue, using the newly available space. If the spool file has no space available, you may respond to the message with an option that will close the printer file in the spool file. When the data in the newly closed printer file has been printed, the spool file space it used is made available again. Processing of the printer file continues from the point at which it was closed, reusing the spool file space. If the spool file frequently becomes full, you should consider increasing the size of the spool file.

Intercept Buffer

Whenever one spool intercept routine intercepts a new printer file, it assigns an intercept buffer in which to accumulate print data and a spool file segment to write the print data into. The data in the intercept buffer is written into the spool file when it becomes full and the buffer can then be filled with more data.

Intercept buffers are assigned from the system assign/free area and returned to it when the printer file is closed. The amount of assign/free area assigned for each printer file depends on the following:

- For a printer file created by a program which has no other spooled printer files currently open, a buffer of 512 bytes will be used if there is sufficient assign/free area. If there is insufficient area, or eight or more programs with printer files open a buffer of 256 bytes will be used.

- For any printer files created by a program having at least one other spooled printer file open, a buffer size of 256 bytes is used.

In addition to the intercept buffers the spool intercept routine itself and other related data areas increase the amount of main storage that the SSP occupies by about 1 K bytes. The system operator should select a system assign/free area size that will accommodate the expected spooling activity.

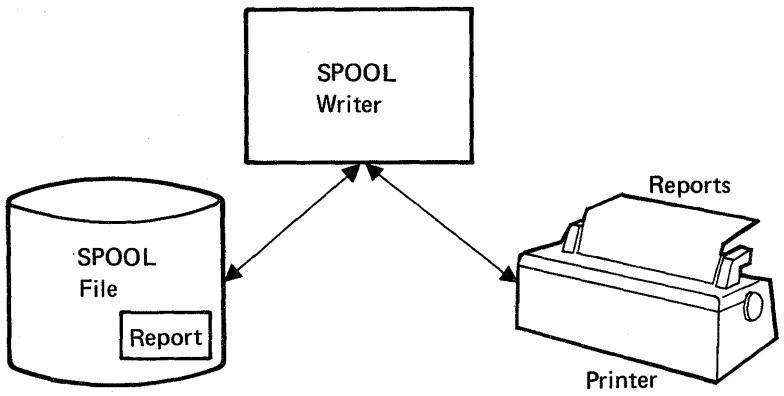
Spool Writer Program

The spool writer program retrieves print data from the spool file and prints it. There is a separate spool writer program for each printer in the system.

Before a spool writer can begin printing, the spool writer program must first be started. If the autowriter function is requested during system configuration or by the IPL overrides, all spool writer programs will be started automatically when you IPL the system. If the autowriter function is not requested, the START PRT command must be entered by the system operator or subconsole operator to start the spool writer programs.

Once a spool writer program has been started, the SSP automatically loads it into main storage to begin printing whenever a printer file is available from the spool file. When nothing is left in the spool file for the writer to print, the writer terminates. When a change is made to the spool file so that another print file becomes available for printing, the spool writer program is again automatically loaded to begin printing.

The following diagram shows how the spool writer program prints jobs from the spool file.



The system operator or subconsole operator may enter a STOP PRT command to stop the spool writer(s). When a spool writer is stopped, it cannot print even if printer files are available for printing. The START PRT command must be entered to allow the writer to print again after it has been stopped.

Printer files are placed in the spool file according to the PRIORITY parameter of the PRINTER OCL statement. The spool writer for each printer selects printer files for printing on a particular printer in order of decreasing priority. Printer files having equal priority are selected according to the order in which they were placed in the spool file. Printer files that are either held, being copied by the COPYPRT procedure or still being intercepted are bypassed and cannot be printed until that condition is changed. (Printer files can be printed while they are still being intercepted if you specify the DEFER-NO parameter on the PRINTER OCL statement or entering the CHANGE DEFER command.)

Changing Forms

The operator and subconsole operator can reduce the number of requested forms changes by specifying a forms number on the START PRT command when starting the spool writer. This causes the spool writer to print only those printer files that require the specified forms. When all files have been printed, the START PRT command should be entered again with either a different forms number, or without a forms number if the writer is to resume printing all printer files.

Before printing each printer file, the spool writer ensures that the forms are positioned at the top of a page by advancing them to line one unless they are already at that position. Not advancing the forms if they are already at line one prevents the spool writer from inserting a blank page between printer files. However, if the previous printer file ended by printing on line one of a page and did not move the forms afterwards, the following printer file will begin printing on that same page and possibly overlay the information previously printed on line one. To avoid this situation, ensure that either your programs do not print on line one, or that they move the forms forward.

Forms Alignment

If forms alignment was requested by the printer file, either in the program that created the file or by the ALIGN-Yes parameter of the PRINTER OCL statement, the spool writer will print the first line of output and then issue a message requesting the operator to align the forms. When the forms have been aligned, the operator may respond to the message, instructing the spool writer to reprint the same line and halt again, or to print the next line and halt again, or to resume normal printing.

Separator Pages

The spool writer will print separator pages ahead of each printer file if desired. The number of separator pages (0-3) may be specified during system configuration for each printer individually, and may be changed by the CHANGE SEP command. If separator pages are either specified during configuration or set to a nonzero value by the CHANGE SEP command, the spool writer issues a message asking the operator whether separator pages should be printed whenever any of the following situations occur:

- When the first printer file is to be printed after the system is IPLed
- When the first printer file is to be printed after the spool writer has been restarted via the RESTART PRT command
- Whenever a printer file is to be printed that requires forms other than those currently in the printer

The spool writer continues printing or not printing separator pages as specified until the operator informs it to do otherwise.

The RESTART PRT Command

If the operator stops the spool writer while it is printing a printer file, entering the RESTART PRT command causes the writer to begin printing that printer file again. A page number may also be entered with the command to inform the spool writer of what page it is to begin printing on. Entering the RESTART PRT command while the spool writer is printing a printer file causes it to start printing the file again from the beginning or from a specific page.

Message Options

The options allowed on the messages that the spool writer issues are generally sufficient for the operator to inform the writer of the action that is to be taken. However, when the writer should take some action other than what is indicated by the message options, the operator may enter the appropriate command to inform the spool writer of the desired action, rather than responding to the message.

Performance Considerations

During system configuration, or through the CHANGE RES command, you can specify whether individual spool writer programs are to be swappable or resident when loaded into main storage. Swappable spool writers may be swapped to and from disk in order to share main storage with other programs running at the same time; resident spool writers do not have this capability. Making a spool writer resident can improve its performance somewhat, because a resident spool writer is not swappable. However, doing this can significantly reduce the amount of main storage that can be used by other programs, as spool writers each require 8 K bytes of main storage.

Another performance factor of the spool writer is that of priority. During system configuration, or through the CHANGE PRTY command, you can specify whether an individual spool writer program is to have normal or high priority when loaded into main storage. A spool writer program with high priority executes before a program with normal priority. The performance of a spool writer program can generally be improved by selecting high priority rather than normal.

In addition to the configuration option, the priority of a spool writer may be changed by the CHANGE PRTY command.

A final factor related to the performance of the spool writer for the 5211/3262 Printer is the spool writer print buffer size. A buffer size of 1 to 4 half-K bytes may be selected. A buffer size of one half-K (512 bytes) is usually sufficient, but in a heavily loaded system, a larger buffer size will generally improve performance.

Spool Commands

Spool commands are the operator's means of controlling both the spool writers and the printer files in the spool file. The following general rules apply to the spool commands:

- Commands entered from the system console can apply to any spool writer or to any printer in the spool file.
- Commands entered from a subconsole apply only to spool writers for printers controlled by the subconsole or to printer files that are to be printed on printers controlled by the subconsole.
- Commands entered from a display station apply only to printer files created by the display station operator.

The following spool commands are used on the System/34:

CANCEL PRT	Changes the priority printing sequence of a specified printer file in the spool file.
CHANGE PRTY	Changes the priority of a specified spool writer.
CHANGE RES	Changes the resident/swappable attribute of a specified spool writer.
CHANGE SEP	Changes the number of separator pages printed by a specified spool writer.
HOLD PRT	Holds selected printer files to prevent them from being printed.
RELEASE PRT	Releases selected printer files that were previously held to allow them to be printed.
RESTART PRT	Restarts a specified spool writer.
START PRT	Starts a specified spool writer.
STATUS PRT	Displays the status of the spool writers.
STOP PRT	Stops a specified spool writer.

After entering a command, the operator is issued a message indicating whether the command was successfully executed.

For more information about these commands refer to the *Operator's Guide* or to the *Command and OCL Statements Reference Summary*.

Identifying Your Spool Output

Output for each printer has a unique six-character identification consisting of the characters SP followed by four numbers, such as SP0042.

You can obtain the status of each job in the spool file by entering D P (Spooled Print Status) command from either the system console, a subconsole, or a work station. The following example illustrates the spooled print status screen.

```
SPOOLED PRINT STATUS                    **COMPLETE**                  CONSOLE    W1
                  BLOCKS AVAILABLE:  1106 OF  1134                    ---PAGES---
 POS ID      PROC      JOBNAME  USER     PRINTER  ID  PRTY  FORM  COPY  TOTAL  WRT
   1 SP0042            W1110559 RON      PRINTKEY P1  A1    0001  1         1    1
   2 SP0043            W1110643 RON      PRINTKEY P1   1    0001  1         1
   3 SP0044            W1110655 RON      PRINTKEY P1   1    0001  1         1




ENTER F-FORWARD, I-INPUT, R-RESTART, U-UPDATE, OR E-END..................... F
```

For more information about the print status screen, refer to the *Operator's Guide*.

The COPYPRT Command

The COPYPRT command executes two programs, $UASF and $UASC, to do the following:

- $UASF copies one or more spool entries into a disk file.
- $UASC displays the contents of the disk file at the work station.
- Both $UASF and $UASC can copy one or more spool entries into a disk file and then display these entries at the display station.

The disk file can be saved on diskette and restored into the spool file for later printing.

Using the STATUS PRT and COPYPRT Commands

After using STATUS PRT to display spool entries in the spool file, you can use the COPYPRT command to copy entries into a disk file according to criteria such as spool ID or forms ID.

The following two examples show the use of the STATUS PRT and COPYPRT commands.

```
SPOOLED PRINT STATUS                    **COMPLETE**                    CONSOLE    W1
                BLOCKS AVAILABLE:  1106 OF  1134                        ---PAGES---
 POS ID      PROC      JOBNAME  USER     PRINTER  ID  PRTY  FORM  COPY  TOTAL  WRT
   1 SP0042            W1110559 RON      PRINTKEY P1  A1    0001  1         1    1
   2 SP0043            W1110643 RON      PRINTKEY P1   1    0001  1         1
   3 SP0044            W1110655 RON      PRINTKEY P1   1    0001  1         1




ENTER F-FORWARD, I-INPUT, R-RESTART, U-UPDATE, or E-END.....................  I
COPYPRT SP0043, COPYFILE, CANCEL, CRT
```

This example shows the copying of a specific spool ID (SP0043). The first entry in the file (SP0042) cannot be copied because it is being printed by the spool writer.

```
SPOOLED PRINT STATUS                    **COMPLETE**                    CONSOLE    W1
                BLOCKS AVAILABLE:  1106 OF  1134                        ---PAGES---
 POS ID      PROC     JOBNAME  USER    PRINTER  ID  PRTY  FORM  COPY  TOTAL  WRT
   1 SP0042          W1110559 RON     PRINTKEY P1  A1    0001  1         1    1
   2 SP0043          W1110643 RON     PRINTKEY P1   1    0001  1         1
   3 SP0044          W1110655 RON     PRINTKEY P1   1    0001  1         1




ENTER F-FORWARD, I-INPUT, R-RESTART, U-UPDATE, OR E-END..................... 1
COPYPRT F0001, COPYFILE, CANCEL, CRT
```

This example shows the copying of all eligible entries with a form number of 0001.

Using Procedure Members and the COPYPRT Command

A procedure member is a library file that contains commonly used control statements. You can place the COPYPRT command into a procedure member and then type the name of the procedure member whenever you want to run the COPYPRT command.

To create a procedure member, you use Source Entry Utility (SEU).

The sample procedure member called SPOOLCPY contains OCL to:

- Determine if a copy file is present.
- Delete the copy file if one exists already.
- Prompt for the digits of the spool ID to be copied.
- Copy the spool entry and display the entry at the display station.

```
* PROCEDURE TO COPY SPOOL JOB TO DISK AND VIEW ON DISPLAY SCREEN
// *  'ENTER SPOOL ID TO COPY'
// IF DATAF1-CPY?WS? DELETE CPY?WS?,F1  IS COPY FILE PRESENT?
COPYPRT SP?1R?,CPY?WS?,CANCEL,CRT
*  ----------END OF PROCEDURE MEMBER----------
```

Related Spooling Documentation

The following list should help you identify System/34 manuals that also contain information concerning spooling:

- The *Installation and Modification Reference Manual* contains information about spooling options specified during system configuration and during IPL.
- The *SSP Reference Manual* describes the PRINTER OCL statement in detail. Chapter 3 also briefly describes operator control commands used for spooling; however, the information in Chapter 3 is intended only to identify for the programmer the operator control commands and their functions. It should not be used as a guide for controlling spooling.
- The *Operator's Guide* contains the information required to actually control spooling from the console or from the subconsole. When you need complete operating information, refer to this manual.
- The *Command and OCL Statements Reference Summary* shows the formats and briefly describes the functions of all operator control commands.

Libraries

A library is a special kind of disk file that contains named groups of data. These named groups are called library members. A library member occupies physically contiguous sectors within a library. Two members cannot share a sector.

TYPES OF LIBRARY MEMBERS

A library can contain four types of members:

- Load members (O members). Load members usually contain instructions that the system can execute. Link-edited load members generated by System/34 compilers and display screen formats created by the $SFGR utility program are examples of load members.

- Procedure members (P members). Procedure members contain collections of frequently used control statements. You can use the $MAINT utility program or SEU to create procedure members in a library. Also you can use a key entry device such as a 3741 Data Entry Station or a 5280 Distributed Data System to create a member as a diskette file and then use the TOLIBR procedure to copy the file into the library.

  For detailed information about the statements and expressions that can be coded within a procedure, refer to *Writing and Using Procedures* in the *SSP Reference Manual.*

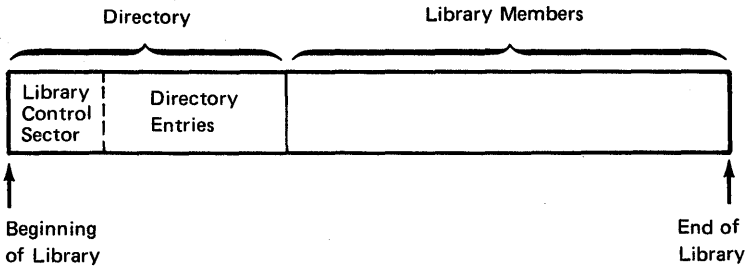
- Subroutine members (R members). Subroutine members are members that must be link-edited (joined) before System/34 can execute them. Some members created by the COBOL compiler, FORTRAN compiler, or the Basic Assembler are examples of subroutine members. BASIC programs can be saved in subroutine member form.

- Source members (S members). Source members contain records such as source specifications or program statements that are used as input to a program or compiler. You can use the $MAINT utility program or SEU to create source members in a library. Also, you can use a key entry device such as a 3741 Data Entry Station or a 5280 Distributed Data System to create a member as a diskette file and then use the TOLIBR procedure to copy the file into the library.

Note: The REPLACE and SAVE commands for BASIC can be used to save a BASIC program as either a subroutine member or a source member. For more information, refer to the *BASIC Reference Manual.*

LIBRARY FORMAT

A library has the following format:



Directory

The directory contains a library control sector and an entry for each member in the library. The minimum directory size is two sectors, and the maximum size is 256 sectors. The first sector in the directory is the library control sector. Each of the remaining sectors can contain nine entries, except for the last sector, which can contain only eight entries. If you know the maximum number of members that you will place in the library, you can determine the required length of the directory.

The library control sector contains a record of the used and available library locations. Each time you add a member or delete a member from the library, the SSP rewrites the library control sector.

The remaining sectors in the library directory contain entries for each of the members in the library.

LIBRARY SIZE

The maximum size for a library is 6553 blocks. This size includes library members and the directory. The maximum directory size is 256 sectors, including the sector used for the library control sector. You can change the size of the system library and system library directory by running the BACKUP and RELOAD procedures. You can change the size of the system library or decrease the size of the system library directory by using the allocate function of the $MAINT utility program. You can change the size of any other library or the size of its directory by using the allocate function of the $MAINT utility. You can also change the size of any other library through the following procedure:

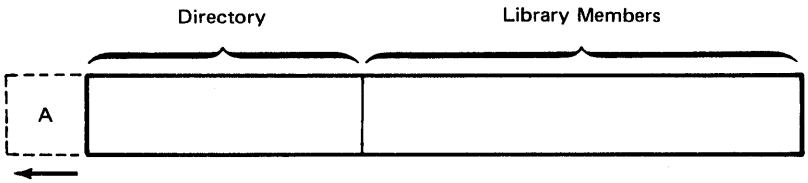
1. Create a new library with more space and a name different from the old library name.

2. Copy the contents of the old library into the new library using the LIBRLIBR procedure.

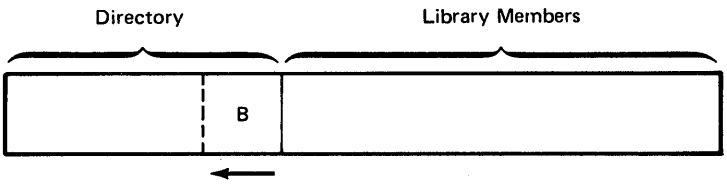
3. Delete the old library.

4. Rename the new library with the name of the old library.

Increasing the size of a user library directory uses disk space that precedes the library. Therefore, disk space, (A) in the following diagram, must be available in order to increase the directory size.

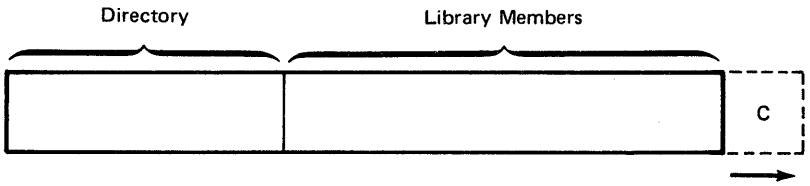


Decreasing the size of a library directory makes disk space, (B) in the following diagram, available for library members.

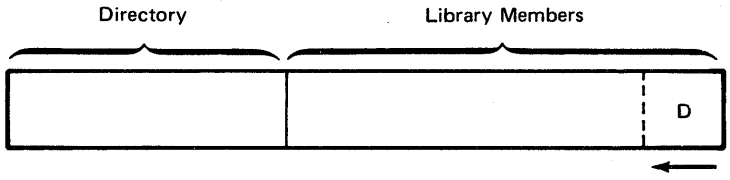


Disk space, (B), must be unused in order to decrease the directory size. The library must be condensed in order to use this directory space for library members.

Increasing the size of a library adds disk space to the end of the library, (C). This disk space must be unused in order to expand the library member area.



Decreasing the size of a library frees space at the end of the library, (D). This library space must be unused in order to decrease the library member area.



REUSE OF LIBRARY SPACE

A library member can reuse the sectors that were occupied by a deleted library member only if *both* of the following conditions are true:

- The job trying to reuse the space is the only job that is using the library, or the system is in single-program mode.
- The deleted member was the last member in the library.

A new version of a library member can be copied into the space occupied by the existing member only if *all* of the following conditions are met:

- The job that is copying the new version of the member into the library is the only job using the library, or the system is in single-program mode.
- The new version fits into the space that is used by the existing member.
- The new version is being copied from a sector-mode file (refer to *Storing Library Members in Disk or Diskette Files* later in this section), or the member was created by one of the following IBM System/34 programs that allow library members to reuse library space:
  - RPG II Compiler
  - Source Entry Utility
  - Assembler
  - Overlay Linkage Editor
  - Work Station Utility
  - FORTRAN
  - $MAINT SSP Utility
  - COBOL
  - BASIC
- The existing member is not an SSP load member in the system library, #LIBRARY.
- The existing member is not a screen format load member.

Because of the conditions that control the reuse of library space, deleting or replacing members can create unusable gaps in a library. The CONDENSE procedure or the compress function of the $MAINT utility program collects this unused space at the end of the library and, thereby, makes it usable for new members. The CONDENSE procedure requires that no active users be signed onto the library. The CONDENSE procedure cannot be run while in the inquiry mode.

ACTIVE USER LIBRARY

Each display station session on System/34 can have an active user library associated with it. For most functions performed during execution of user programs, the SSP first searches the active user library for required library members. If a required member is not in the active user library, the SSP searches the system library, #LIBRARY. Functions for which the SSP first searches the active user library are:

- Loading a program
- Invoking a procedure
- Displaying a menu or a display screen format
- Using a message member (MEMBER OCL statement)

For most functions performed during execution of utility programs or compilers, the SSP searches only the system library, #LIBRARY, unless you specify a different library. Functions that do not automatically use the active user library are:

- Retrieving records from a message member. The SSP searches only the library that was active when the MEMBER OCL statement was entered.
- Maintaining libraries using the $MAINT SSP utility program.
- Using DFU, SEU, WSU, or SDA.
- Using RPG II, FORTRAN, COBOL, or Basic Assembler.
- Using BASIC. (Every library used during a BASIC session is active until the BASIC session ends.)

The operator can specify the active user library during session sign on and can change the active user library for the session by entering the LIBRARY OCL statement, by using the SET procedure, or by specifying a library name on the BASIC or BASICR procedure. The active user library can be changed for the duration of a procedure via a LIBRARY OCL statement within the procedure.

LIBRARY SHARING

On System/34, two or more programs can read from or write to the same library concurrently. However, if one program writes to a library while another uses the same library or if two or more programs concurrently write to the same library, performance might be significantly degraded.

Certain library functions require that no other functions use the same library concurrently. These library functions are:

- Compressing a library via the CONDENSE procedure.
- Deleting all library members or deleting all members of one type via the REMOVE procedure.
- Reusing library space. Refer to *Reuse of Library Space* earlier in this section for further information.

STORING LIBRARY MEMBERS IN DISK OR DISKETTE FILES

Library members can be stored in disk or diskette files in *record mode* or *sector mode*. A record-mode file on diskette can be used on other systems if it is written as a basic data exchange file. A sector-mode file on diskette can be used on another System/34.

Record-Mode Files

Source and procedure members can be stored as record-mode files. Records in a record-mode file can be from 40 to 120 characters long, but all records in a file must be the same length. The SSP pads records with blanks (hexadecimal 40s) or truncates records to the specified record length.

The first record in a member stored in a record-mode file on disk or diskette is a COPY statement that defines the attributes of the member. The last record in a member stored in record-mode file is a CEND statement. When you use the $MAINT utility program to copy a source or procedure member into a record-mode file, $MAINT places the COPY and CEND statements in the file. If you use SEU, your own program, or a key entry device to create a record-mode file, you must place the COPY and CEND statements in the file. If the record-mode file is organized as a direct file, you must also include an END statement following the CEND statement that terminates the last member in the file. The formats and descriptions of the COPY, CEND, and END statements are included in the descriptions of record-mode files in the *SSP Reference Manual*.

Note: When you use $MAINT to save a procedure member in a record-mode file, you should specify SVATTR-YES if you also want the procedure attributes saved. Examples of procedure attributes are MRT-YES, PDATA-YES, and HIST-YES.

Sector-Mode Files

Any library member can be stored as a sector-mode file. All members copied into a sector-mode file have 40 bytes of control information preceding the member. This control record contains 28 bytes for the member's directory entry, 8 bytes of PTF information that is used for diagnostic information, and 4 reserved bytes.

If you copy a System/32 sector-mode file to a System/34 library:

- System/32 SCP members are not copied to the System/34 library.
- The SSP sets the release level of the copied members to 232 to indicate that the members came from a System/32.
- Source or procedure members copied from the sector-mode file may require more library space than they did on System/32. More space might be required because System/32 and System/34 use different blank character compression schemes.

Saving a Library on Diskette

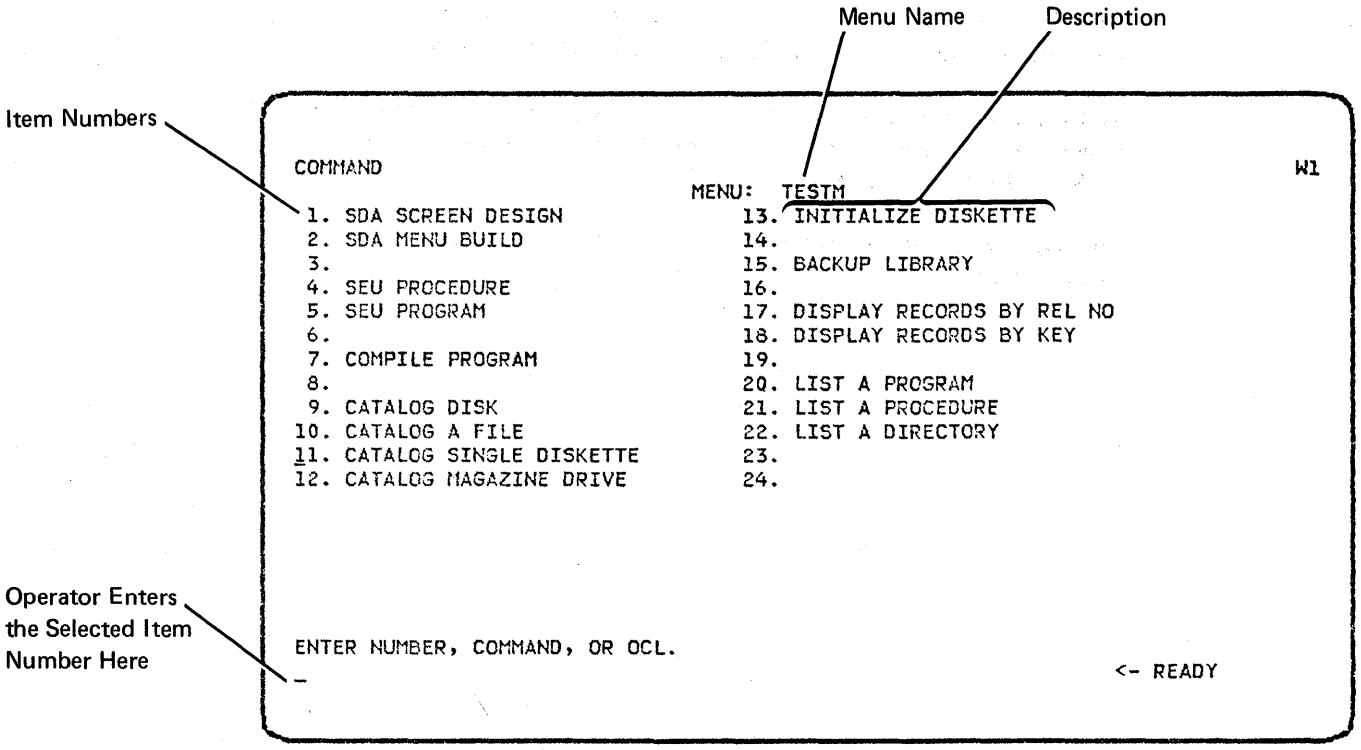
You can save your user libraries by using the SAVELIBR procedure. The SAVELIBR procedure copies all library members (except SSP members) from a user library to a diskette file. BLDLIBR or TOLIBR must be used to restore the library you have saved.

Notes:

1. System/32 diskette files created by releases 1 through 4 of System/32 cannot be read by System/34 until they are updated by the System/32 CONVERT procedure.
2. You can copy subroutine members (R members) or load members (O members) from System/32. System/32 load members cannot be run on System/34. For example, SEU and DFU members can be copied from System/32 to System/34, but the object message members for them cannot be used on System/34. You should test subroutines carefully because you might have to recompile them for use on System/34.
3. You cannot copy a System/34 sector-mode file to a System/32 library.

Menus

A menu is a programmer-defined list of item numbers and brief descriptions that appears on the display screen at a display station operator's request or when the operator signs on if sign-on menu security is active. When the operator selects an item number, the SSP does the function associated with that number. The following is a sample menu:



Menus simplify the operator's duties. The operator needs no knowledge of the OCL statements, procedure commands, or operator control commands required to do a function.

The operator displays a menu by:

- Entering a menu name during signon. The requested menu appears as soon as signon ends.
- Signing on, if sign-on menu security is active and the operator has a menu assigned to him.
- Entering a MENU operator control command.
- Selecting a menu from another menu, which is called menu chaining.

The programmer can cause a menu to be displayed at the end of a job by using the MENU OCL statement within the job.

Each menu has an associated program-defined command load member. When the operator enters a menu item number, the SSP retrieves the record corresponding to that item number from the command load member. The command load member is created when the menu is created. The retrieved record can be an OCL statement, a procedure command, or an operator control command. The SSP continues job processing just as if the operator had entered the retrieved record from the keyboard. For further information about job processing, refer to *System/34 Job Processing* earlier in this chapter.

For information about creating menus, refer to the *IBM System/34 SDA Reference Manual* and the *IBM System/34 SSP Reference Manual.*

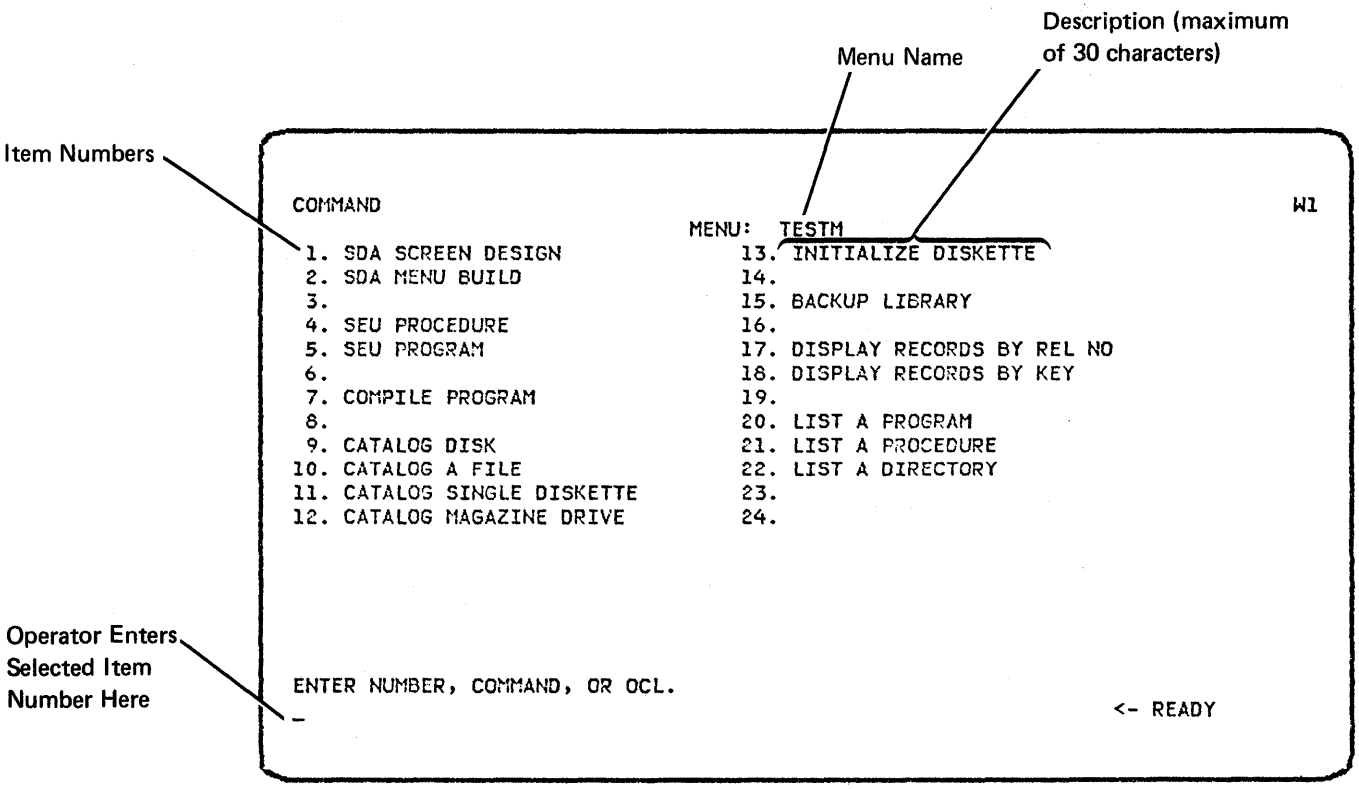
To leave a menu display when sign-on menu security is not active, an operator enters a zero instead of an item number. The menu disappears, and the COMMAND display appears.

To end a menu display when sign-on menu security is active, the operator must sign off.

FIXED-FORMAT AND FREE-FORMAT MENUS

System/34 provides two types of formats for menus: fixed-format and free-format. A fixed format menu contains 24 numbered items. The description of each item can have as many as 30 characters, including blanks. As the following displays show, a fixed-format menu fits on one 1920-character display screen and on two 960-character display screens. When using a 960-character display screen, the operator presses the Enter/Rec Adv key to select the second half of the menu.

Fixed-Format Menu for a 1920-Character Display Screen



Fixed-Format Menu for a 960-Character Display Screen

```
COMMAND                                                          W1
                                        MENU: TESTM
 1. SDA SCREEN DESIGN                       13. INITIALIZE DISKETTE
 2. SDA MENU BUILD                          14.
 3.                                         15. BACKUP LIBRARY
 4. SEU PROCEDURE                           16.
 5. SEU PROGRAM                             17. DISPLAY RECORDS BY REL NO
 6.                                         18. DISPLAY RECORDS BY KEY

ENTER NUMBER, COMMAND, OR OCL.
_                                                          <- READY
```

```
COMMAND                                                          W1
                                        MENU: TESTM
 7. COMPILE PROGRAM                         19.
 8.                                         20. LIST A PROGRAM
 9. CATALOG DISK                            21. LIST A PROCEDURE
10. CATALOG A FILE                          22. LIST A DIRECTORY
11. CATALOG SINGLE DISKETTE                 23.
12. CATALOG MAGAZINE DRIVE                  24.

ENTER NUMBER, COMMAND, OR OCL.
_                                                          <- READY
```

A free-format menu on a 1920-character display screen allows the programmer to completely define lines 3 through 20. A free-format menu on a 960-character display screen allows the programmer to completely define lines 3 through 8 of two display screens. The description of each item is not limited to 30 characters. The operator can press the Enter/Rec Adv key to select the second half of the menu. The following displays show free-format menus on the 1920-character and 960-character display screens.

Free-Format Menu on a 960-Character Display Screen

```
COMMAND
                              MENU: TESTM

                        1. SDA SCREEN DESIGN
                        2. SDA MENU BUILD
                        3.
                        4. SEU PROCEDURE
                        5. SEU PROGRAM
                        6.
ENTER NUMBER, COMMAND, OR OCL
_                                                          <- READY
```

```
COMMAND
                              MENU: TESTM

                        7. COMPILE PROGRAM
                        8.
                        9. CATALOG DISK
                       10. CATALOG A FILE
                       11. CATALOG SINGLE DISKETTE
                       12. CATALOG MAGAZINE DRIVE
ENTER NUMBER, COMMAND, OR OCL
_                                                          <- READY
```

If a display station has a 960-character display screen and a free-format menu for a 1920-character display screen is selected, lines 3 through 8 of the menu are displayed. If the Enter/Rec Adv key is pressed without any data being entered, lines 9 through 14 are displayed.

Lines 15 through 20 of the menu cannot be displayed on the 960-character display screen.

Refer to the *IBM System/34 SDA Reference Manual* and the *IBM System/34 SSP Reference Manual* for further information about fixed-format and free-format menus.

Program Attributes

Program attributes describe a program's use of display stations or use of resources on System/34.

Attributes that can be specified when a program is compiled are:

- SRT (Single Requestor Terminal). The program allows one requesting display station and can acquire data display stations. Display stations can be defined as data display stations during system configuration, or a command display station can be placed in data mode via the MODE control command.

- MRT (Multiple Requestor Terminal). This attribute allows more than one requesting display station for a single copy of the program. The program can acquire data display stations.

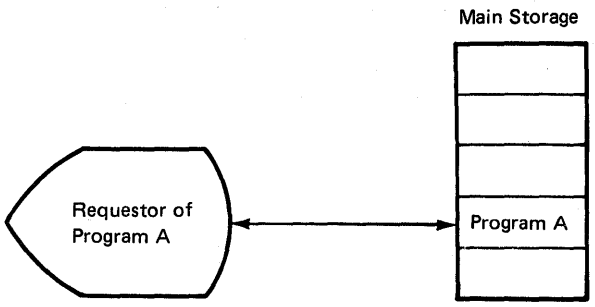
- NEP (Never-Ending Program). This attribute can be given to SRT programs and MRT programs. Programs do not wait for nonshared resources that the NEP uses. An MRT-NEP may wait for additional requestors when no requestors are attached to it.

  Note: The STOP system command removes the NEP attribute from an MRT-NEP program.

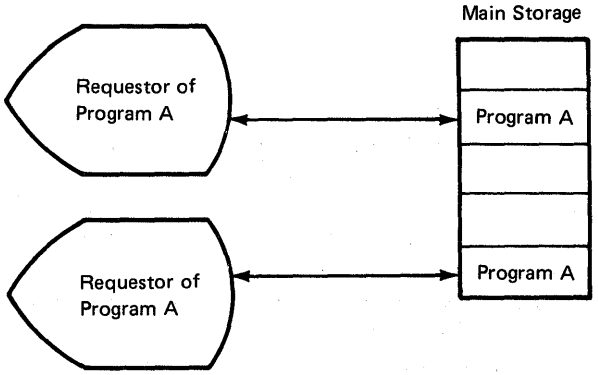
This section describes the concepts of each of these attributes. For information about choosing an attribute during program design, refer to *Application Design* in Chapter 3. For information about jobs that are not attached to a requesting display station, refer to *Jobs That Run Without a Requesting Display Station*, later in this chapter.

SRT (SINGLE REQUESTOR TERMINAL) PROGRAM

An SRT program is a program that allows only one requesting display station. Each time the program is requested, that program is loaded into main storage. The display station that requests the program becomes attached to that program.



If two display stations request the same SRT program, two copies of the program will be in main storage.



Each requestor has its own local data area and external indicators which can be used to pass data to the program.

Coding SRT Programs

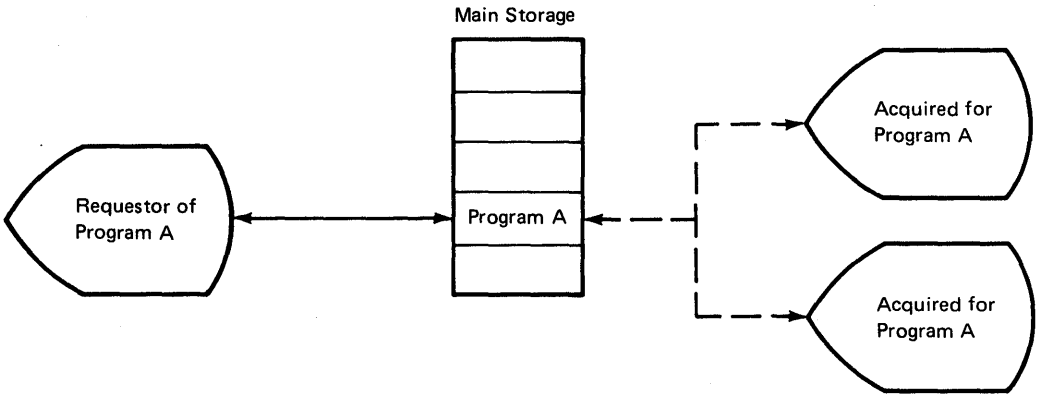
The SRT attribute is given to a program when it is compiled. A program will be an SRT program if the MRTMAX parameter is zero or is omitted from the COMPILE OCL statement. Refer to the *SSP Reference Manual* for a description of the COMPILE OCL statement.

RPG II, COBOL, FORTRAN, BASIC, and Basic Assembler programs can be SRT programs. WSU programs are always generated as MRT programs and cannot be changed to SRT programs.

Acquiring a Display Station in an SRT Program

An SRT program can acquire other data mode display stations, which are used to enter data and/or display output. A display station can be acquired in either of the following ways:

- A display station is acquired if the job step that executes the SRT program includes a WORKSTN OCL statement. If the REQD-YES parameter is specified on this statement, the display station is attached to the program by the system, and the display station must be available (signed on and in standby mode) so that the program can be initiated.



- If the REQD parameter is NO or omitted from the WORKSTN OCL statement, an RPG II program or a COBOL program, instead of the system, can acquire the display station via an operation code. An acquired display station cannot be operated in inquiry mode. If a command display station interrupts the program, the acquired display stations are inactive during the inquiry request. After acquiring a data display station, the System/34 data management handles it the same as a command display station. The SRT program logic should be the same as the logic required for an MRT program.

Releasing Display Stations from an SRT Program

A display station acquired by the system or by an operation code within an SRT program can be released by the program. For example, for RPG II programs a display station is released via the REL operation code or via an R entry in column 16 of the RPG II output specification. For COBOL programs, a display station is released via the DROP operation code.

The requesting display station cannot be released from an executing SRT program until that program goes to end of job. End of job can occur only when all acquired display stations have completed processing and have been released. Therefore, the requesting display station cannot be released until all acquired display stations have been released.

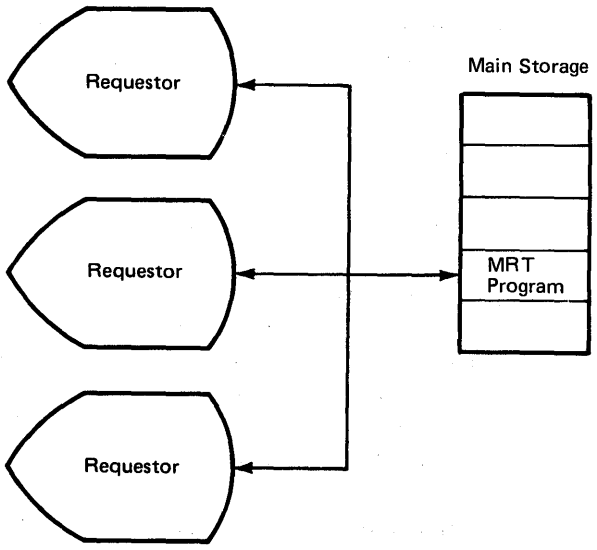
Interrupting an SRT Program

An SRT program can be interrupted only from the display station that requested the program. The interruption is called an inquiry request and is caused by the operator pressing the Attn key. The SRT program stops executing, and the Inquiry display appears. Refer to the System/34 *Operator's Guide* or *SSP Reference Manual* for a description of the Inquiry display. The display station operator can select one of the following options from the Inquiry display:

Option	Function
0	Resume execution of the interrupted SRT program.
1	Continue with the inquiry request. The display station enters command mode. If the interrupted program is an RPG II program, option 1 appears only if the H specification permits inquiry (column 37 contains a B).
2	End the interrupted SRT program and close the files.
3	End the interrupted SRT program and do not close the files.
4	Post an inquiry latch, or terminate a BASIC program and return to BASIC command mode. An interrupted RPG II program can check the inquiry latch by using the RPG II subroutine, SUBR95. For information about SUBR95, refer to the *RPG II Reference Manual*.
5	Display the session status.

MRT (MULTIPLE REQUESTOR TERMINAL) PROGRAM

An MRT program is a program that allows several requesting display stations to be attached to one copy of the program at a time.



An MRT program can be called only from an MRT procedure. An MRT program cannot be called by OCL statements entered from the keyboard.

An MRT procedure can be called from either the keyboard or a non-MRT procedure via an INCLUDE OCL statement or a procedure command.

When an MRT procedure is called, the SSP checks whether the requested procedure is active. If the procedure is not active, the SSP loads and initiates the MRT program. If the procedure is active and no more than the maximum number of display stations are using the program, the requesting display station is attached to the program. If the procedure is active and the maximum number of display stations is attached to the program, the SSP places the display station on a waiting-for-resources queue. Refer to *Never-Ending Program* later in this chapter for further information about programs waiting for system resources.

OCL statements in the MRT procedure are executed for the requestor that actually initiates the MRT program. Subsequent requestors bypass the OCL processing and attach directly to the MRT program.

For RPG II MRT programs, a display station attaches to the program at the beginning of an input cycle or at the last READ operation from the WORKSTN file; for WSU programs, a display station attaches at the work-session-initiation processing level. For COBOL programs, a display station attaches at the last read or call operation that executes.

Ordinarily, two different MRT procedures should not call the same MRT program. If two such procedures are active at the same time, two copies of the same MRT program will execute concurrently. When response times are long because of many display stations using the same MRT procedure, you might allow two MRT procedures, each allowing half the total number of requestors, to call the same program.

For WSU programs, an MRT procedure is generated when the WSU program is generated. For RPG II, COBOL, Basic, and Assembler programs, the programmer must provide the MRT procedure. If SEU is used to create this procedure, SEU prompts for the MRT attribute at the end of the SEU run. If $MAINT is used to create this procedure, the MRT-YES parameter of the COPY statement allows specification of the MRT attribute.

The following are additional facts about MRT procedures and MRT programs:

- Only one LOAD OCL statement and one RUN OCL statement are allowed in an MRT procedure. Any statements that follow the RUN OCL statement are ignored.

- An MRT procedure can be called from a higher level procedure but cannot call a lower level procedure.

- All MRT procedure names should be unique, even if the procedures are in different libraries.

- When an MRT procedure is originally requested from another procedure, a new job is (in effect) started on the system. Therefore, OCL statements such as the REGION and ATTR statements within the MRT procedure are processed as if they were at the beginning of a job.

- The OCL statements are not processed again for subsequent requestors.

- The INCLUDE OCL statement or the procedure command statement cannot pass parameters to the MRT procedure. However, the INCLUDE OCL statement or procedure command can pass data to the MRT program. The SSP passes the data to the program on the first input operation from the requesting display station. Data passed to the program starts with the first nonblank character following the procedure name and ends with the last nonblank character in the statement.

- If an attempt is made to call an MRT procedure when the system is in single-program mode, the procedure runs but not as an MRT procedure. Therefore, the SSP treats any data coded on the INCLUDE OCL statement or the procedure command statement as parameters.

- Except for WSU programs, file sharing conflicts *within* an MRT program are the programmer's responsibility. These conflicts are not handled by System/34 file sharing logic. Refer to *File Concepts* earlier in this chapter for further information.

- An MRT program cannot use job (RETAIN-J) files created by previous steps in the job. Subsequent non-MRT job steps can use those job files. In addition, job files (RETAIN-J) created by an MRT program are treated like scratch files. These files are not passed to subsequent job steps, and they are deleted when the MRT program goes to end of job.

- Any DATE, FORMS, MEMBER, PRINTER, or SYSLIST statement that has been used in a previous step has no effect on a job step that runs an MRT program. Instead, the MRT program uses values in the system configuration record.

- When the MRT program releases its last requesting display station, the program might not immediately go to end-of-job processing. Therefore, the MRT program might still be executing while the next statements in the job stream of the requesting display station are being processed. For that reason, an IF ACTIVE test that determines if the MRT procedure has been terminated should not follow the MRT procedure call in the job stream.

- An MRT program can access the requesting display station's local data area and external indicators. For example, RPG II and COBOL provide subroutines to read and modify each display station's local data area and external indicators. COBOL also provides language extensions for retrieving and updating these areas.

Coding MRT Programs

The MRT attribute is given to a program when it is compiled. A program will be an MRT program if the MRTMAX parameter is specified on the COMPILE OCL statement and is not MRTMAX-0. An MRTMAX value of one is valid and means that only one display station can be attached to the program at one time, but that multiple copies of the program are not initiated when more than one display station operator requests the program. Refer to the *SSP Reference Manual* for a description of the COMPILE OCL statement. The maximum number of requestors can be changed when the program is run via the ATTR OCL statement. Program logic should be checked before increasing the maximum number of display stations.

RPG II, WSU, COBOL, and Basic Assembler programs can be MRT programs. WSU programs are always generated as MRT programs and automatically adjust their program size when the maximum number of requestors is changed via the ATTR OCL statement. WSU programs do not have to be regenerated to allow for an increased maximum number of requestors.

BASIC programs can also be MRTs. BASIC programs are not compiled and do not use the // COMPILE OCL statement. Refer to the *BASIC Reference Manual* for details on creating BASIC MRTs.

Acquiring a Display Station in an MRT Program

An MRT program can acquire other data display stations, which are used to enter data and/or display output. A display station is acquired if the job step that executes the program includes a WORKSTN OCL statement. If the REQD-YES parameter is specified on this statement, the display station is attached to the program by the system, and the display station must be available so that the program can be initiated. An available display station is one that is signed on and in standby mode. A data display station can also be acquired by program logic. This logic must determine the actions to take if the display station is not available. BASIC MRT programs use the OPEN statement to acquire other display stations.

Releasing an Acquired Display Station from an MRT Program

A display station acquired by the system or by an operation code within the program can be released by the program. For example, for RPG II programs a display station is released via the REL operation code or via an R entry in column 16 of the RPG II output specification. For COBOL programs, a display station is released via the DROP statement. The display station returns to standby mode after it is released. BASIC MRT programs release the display station by using the CLOSE statement.

Interrupting an MRT Program

A requestor can interrupt his attachment to an MRT program to do other work while allowing the MRT and its other users to continue. The interruption is called an inquiry request and is caused by the operator pressing the Attn key. The Inquiry display appears, and the display station operator can select one of the following options to:

Option	Function
0	Resume use of the MRT program.
1	Continue with the inquiry request. The display station enters command mode.
2	Release the display station from the MRT program and continue processing the next job step in the job being run from the display station.
3	Release the display station from the MRT program and cancel the remaining steps in the job being run from the display station.
5	Display the session status.

Maximum Number of Display Stations for an MRT Program

For RPG II programs, the MRTMAX value on the COMPILE OCL statement, on the ATTR OCL statement, or on the RPG command statement specifies the maximum number of requesting display stations for an MRT program. A continuation line of the WORKSTN file description specification specifies the total number of requesting *and* acquired display stations for an MRT program. The keyword NUM is specified, and the maximum number of display stations is coded in positions 60-65 of the continuation statement.

For COBOL or Assembler programs, the MRTMAX value on the COMPILE OCL statement or on the ATTR OCL statement specifies the maximum number of requesting display stations for an MRT program. Program logic determines the actual maximum number of display stations that are supported. Internal tables in the program might have to be incremented in order to support an increased maximum number of allowed requestors.

For BASIC programs, the MRTMAX value on the ATTR OCL statement specifies the number of requesting display stations for an MRT program.

For WSU programs, the MRTMAX value on the ATTR OCL statement is set to the value specified on the WSU J specification when the program is generated. If this value is changed, WSU automatically adjusts the program to its newly required size without requiring program regeneration.

If a display station requests an MRT program while the program is handling its maximum number of display stations, the SSP queues that display station request to the MRT program. When the MRT program releases one of its requestors, the SSP attaches the first queued requestor to the MRT program. While a display station waits for its request to be honored, the display station cannot be used unless the operator interrupts the program and releases his display station from the MRT procedure.

Releasing Requesting Display Stations from MRT Programs

An MRT program must release a requesting display station when the program has completed processing for that display station. If it is not released, the display station remains allocated to the MRT program until it ends or until the operator interrupts the program and releases the display station.

For example, for RPG II MRT programs, a requestor can be released by the REL operation code or by an R entry in column 16 of output specification. For COBOL programs, a requestor can be released by the DROP operation. For BASIC programs, a requestor can be released by the CLOSE statement.

A released requesting display station returns to the place from which the MRT program was called. This return can be to:

- The next OCL statement in the procedure that called the MRT program
- A menu
- The command display

Ending MRT Programs

An RPG II MRT program that is not an NEP ends when the LR indicator is set on. This occurs when the end of file is reached for the primary input file. A programmer should not arbitrarily end an MRT program by setting on the LR indicator, because requestors might still be attached to the program.

A COBOL or Assembler MRT program that is not an NEP receives a no-outstanding-invites code when no display stations are attached or when no attached display stations have been invited.

A BASIC program ends normally when the last operator using the program ends the session.

The programmer must provide coding to detect the no-outstanding-invites condition and end his program. If a request for the MRT program is made while it is ending, another copy of the MRT program is initiated.

A WSU program ends normally when the last operator using the program ends his session. Also, a WSU program ends if the EJ indicator is on when the processing for a display completes. All attached display stations automatically begin end of work session (EW) processing when they return to the WSU program.

Canceling an MRT Program

Use the CANCEL command from the system console to cancel an MRT program.

Using the Attn Key to Release a Display Station from an MRT Program

You should be aware of the following consideration when using the Attn key to release your display station from an MRT program. The Attn key options do not take effect until your MRT program issues a read command to the display station. If, for example, you have a coding error in your MRT program, such as a loop in your RPG CALC specifications that prevents a read command from being issued, you cannot use the Attn key to release your display station from the MRT program.

NEVER-ENDING PROGRAM (NEP)

The never-ending-program attribute can be assigned to MRT programs, SRT programs, and programs run from the input job queue. This attribute can be assigned to an MRT program to allow it to remain active when all its requestors have been released. The program will probably be swapped out of main storage. New requestors can attach to the program without waiting for program initiation, which can require a significant amount of time. When a new requestor is attached, the program is simply swapped into main storage. (Disk activity requires much of the time used for program initiation. For further information, refer to *Disk Activity for Loading Programs and Attaching Display Stations to Them* in Chapter 3.)

An MRT program without the never-ending-program attribute does not remain active without requestors and is reinitiated for a subsequent request. Therefore, MRT programs that are requested frequently might be assigned the never-ending-program attribute to avoid this reinitiation time.

The never-ending-program attribute can be assigned to an SRT program. Typically, this attribute is assigned to an SRT program that uses nonshared system resources such as a printer or a nonshared disk file. If another program that requires the nonshared resource is requested, the system waits for the SRT program to end its use of the resource before starting the new program. Because the SRT program has a never-ending-program attribute, the operator that requested the new program receives a waiting-for-nonshared-resource message. This message allows the operator to cancel the new program or request that the SSP retry allocating the resource.

If the SRT program had not been never-ending, the new program's operator would receive no indication that the program was waiting for a resource. However, the operator could use the STATUS USERS command to see that the program was waiting for initiation. The Status Active column on the User's Status display contains INIT-WAIT for the new program.

MRT programs and programs run from the input job queue automatically issue the waiting-for-nonshared-resource message to the operator unless NEP-NO is specified for the program.

Figure 2-5 summarizes the effect of specifying the never-ending-program attribute for an MRT program, an SRT program, and a program run from the input job queue.

Program Type	Never-Ending-Program Attribute	Waiting-for-Nonshared-Resources Message	Active when No Requestors
MRT	NEP-YES	Yes	Yes
	NEP-NO	No	No
	Not specified	Yes	No
SRT	NEP-YES	Yes	–
	NEP-NO	No	–
	Not specified	No	–
Input job queue	NEP-YES	Yes	–
	NEP-NO	No	–
	Not specified	Yes	–

Figure 2-5. Never-Ending-Program Attribute Summary

Coding Never-Ending Programs

The NEP attribute is given to a program when it is compiled by specifying NEP-YES on the COMPILE OCL statement. The NEP attribute can be overridden when the program is run by specifying NEP-YES on the ATTR OCL statement. The *System/34 SSP Reference Manual* describes these statements.

A WSU program is not generated as never-ending, but it can be defined as never-ending by modifying the procedure that calls the program. The NEP-NO parameter on the ATTR OCL statement can be changed to NEP-YES to make the program never-ending.

Ending a Never-Ending MRT Program

Typically, an MRT never-ending program is coded to accept input from a display station, process the input, release the requesting display station, and then accept input from any requesting display station. When the program has no requesting display stations, the SSP places the program in a wait condition. If the system operator enters a STOP SYSTEM command, the never-ending program(s) receives control with an indication of the operator's request so that the program can execute end-of-job logic.

An MRT never-ending RPG II program is given control when the STOP SYSTEM control command is entered by the system operator. The program should test for this condition and end the program.

A WSU never-ending program ends either when the EJ indicator is set on in the program or when the program is stopped by the operator.

A COBOL or Assembler never-ending program must be coded to end when the no-outstanding-invites return code is indicated. This return code is indicated only when the STOP SYSTEM command has been entered and the program has released all its display stations.

Jobs That Run Without a Requesting Display Station

Some jobs do not require interaction with a requesting display station. These jobs can execute while the requesting display station continues with other work. You can initiate such a job in one of three ways:

- Placing the job on the input job queue.
- Explicitly releasing the requesting display station, either with the RELEASE-YES parameter on the ATTR OCL statement or with the release operation code in a program.
- Evoking the job with an EVOKE OCL statement or with an evoke-end-of-transaction SSP-ICF operation.

The following considerations apply to jobs that are not attached to a requesting display station:

- Messages are displayed at the system console.
- Changes that such a job makes to external indicators or to the local data area are in effect only during execution of the job. The changes are not accessible to any other jobs.
- Except for jobs run from the input job queue, system list output goes to the system list device that was active when such a job was initiated. System list output for jobs run from the input job queue goes to the system printer or the default printer for released jobs. Printer data management output goes to the system printer, unless a printer default for released jobs was specified during system configuration. (You can, of course, use the PRINTER OCL statement to direct output to an individual printer.) For information about system list output and printer data management output, refer to *Printer Concepts*, earlier in this chapter.
- When the requesting display station is explicitly released, job (RETAIN-J) files created by previous job steps are not available to the released job.
- Jobs on the input job queue are processed one at a time on a first-in, first-out basis within job class. The operator can assign different job classes by using the JOBQ command before placing the job on the queue. In that case, the jobs within a higher job class are started before other jobs within lower job classes. For example, jobs in job class 5 are started by the system before jobs in job class 4.

  The operator can also assign execution priorities to jobs. Jobs with a low execution priority are more likely to be swapped than jobs with a higher execution priority.

- The system operator can start, stop, assign execution priority and job class, cancel, and change the order of jobs in the input job queue. The display station operator can assign execution priority and job class before the job is placed in the queue or can cancel the job after it is in the input job queue.

- For jobs run from the input job queue, the display station environment at the time the job was submitted is saved and used when the job runs. The following information is saved:
  - Printer information: Information such as session printer for the job, the forms number used with the job, and the lines per page specified for the session printer.
  - Identification information: Job name, user ID associated with the job, and the date.
  - Main storage information: The total region size and the job region size.
  - Local data area.
  - System configuration information for batch BSC jobs.

  The following information is not saved when jobs use the input job queue:
  - Communications configuration record.
  - Print belt image. The jobs use the print belt information contained within the configuration record.

An example of a job that runs without an attached requesting display station is a job that prints transactions entered from many display stations.

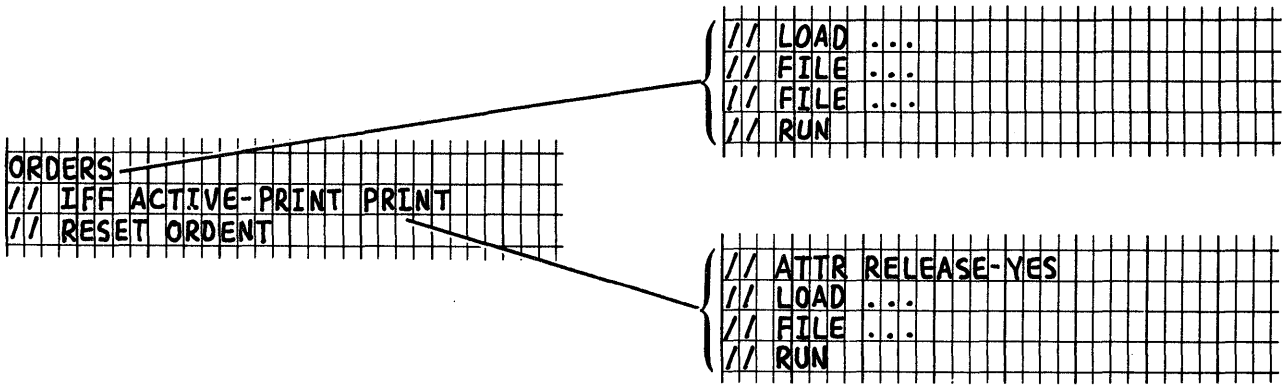
A procedure could be coded to do the following:

1. Load a program to enter the transactions. This program ends when the operator finishes entering transactions.

2. Load a print program to print the transactions. When the program is loaded, its requesting display station is released.

3. Return to Step 1 to allow the operator to enter more orders.

The sample OCL in this procedure, ORDENT, is:

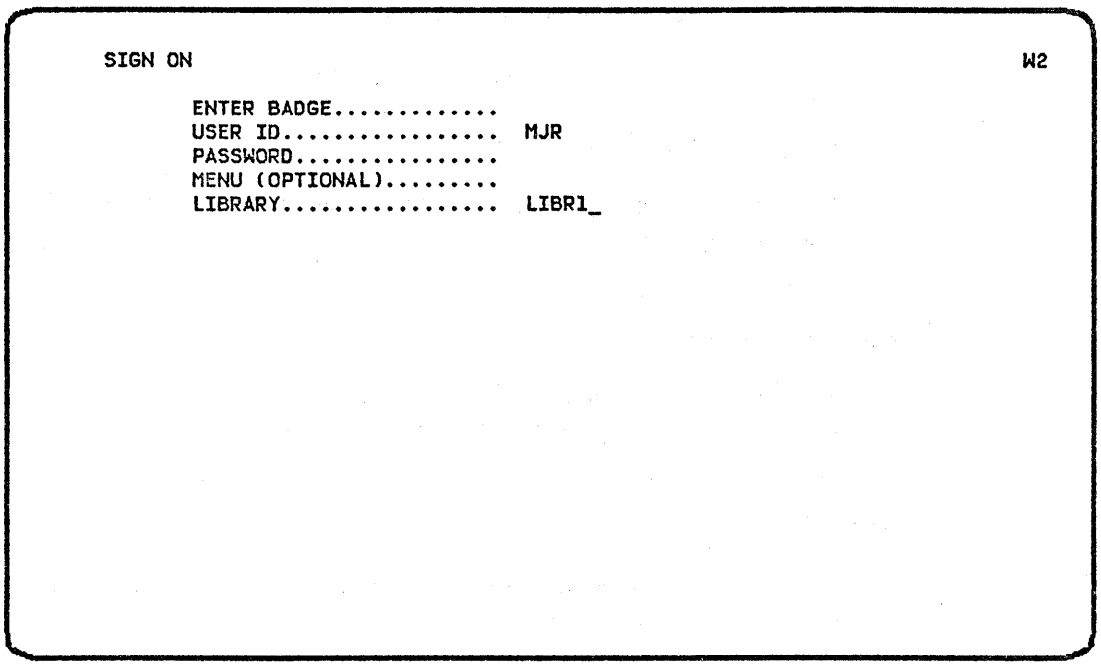


System-Provided Security

System/34 provides sign-on security and file/library security. Sign-on security includes password, badge, and menu security. This section describes major aspects of these types of security. Refer to the *Installation/Modification Reference Manual* for further information about security.

PASSWORD SECURITY

Password security prevents unauthorized use of a display station. To begin a session, an operator must enter his password into a nondisplay field on the sign-on display. Nondisplay means that the entered characters do not appear on the display.



If the operator does not enter the correct password, he cannot begin his session. A session is the time from sign on to sign off that an operator uses a display station.

A disk file called the password security file contains a profile for each person that is authorized to use the system. Each profile contains:

- An 8-character user ID
- A 4-character password assigned to the user
- A badge ID if badge security is used
- Mandatory menu indicator
- A menu name if menu security is used
- A library name (can be the library containing the menu or a default library)
- A code that identifies the user's security class
- Service aid authorization (patch, dump, setdump)
- A 20-character comment field

The password security file can be initialized and updated via the PROF or PRMENU procedure.

Security Classifications

The security classifications are master security officer, security officer, system operator, subconsole operator, and display station operator.

Master Security Officer

The master security officer classification is assigned during the initial definition of password security. This officer can:

- Save and restore the password security file. The $PRSV utility saves the file, and the $PRST utility restores the file. These utilities can be run only by the master security officer.
- Redefine the password security file.
- Add, delete, and edit profiles of security officers, subconsole operators, system operators, and display station operators.
- Change his own security profile information.
- Act as a system operator, subconsole operator, or display station operator.
- Identify files and libraries to be protected and identify the owners of the protected files and libraries.
- Select and remove file/library security, password security, and badge security.

Security Officer

Security officer classifications are assigned by the master security officer. A security officer can:

- Add, delete, or edit profiles of system operators and display station operators.
- Change his own password and badge ID.
- Act as a system operator, subconsole operators, or display station operator.
- Identify the files and libraries to be protected and identify the owners of the protected files and libraries.

System Operator

System operator classifications are assigned by the master security officer or by a security officer. An operator designated as a system operator can operate any display station, including the display station defined as the system console.

Subconsole Operator

Subconsole operator classifications are assigned by the master security officer or by a security officer. An operator designated as a subconsole operator can operate any display station except the display station defined as the system console.

Display Station Operator

Display station operator classifications are assigned by the master security officer or by a security officer. An operator designated as a display station operator can operate any display station, except the display station defined as the system console. If a display station operator signs onto a subconsole, he is not allowed to operate the display station in subconsole mode.

BADGE SECURITY

Badge security can be active only with password security and prevents unauthorized use of a display station. When badge security is active, the sign-on display has an Enter Badge prompt on it. When the sign-on display appears, the cursor is at the Enter Badge field. The operator must move his badge through a magnetic stripe reader, which reads but does not display the badge ID. The operator enters the remaining fields on the sign-on display. In order to sign on, the operator must have used the proper badge and entered the correct password.

The master security officer uses the PROF procedure to activate badge security. He uses the CNFIGSSP procedure to designate which display stations have magnetic stripe readers.

Badge security requires a magnetic stripe reader at each designated display station and the hardware support for connecting the reader to the display station. The badge consists of special encoding of data onto a magnetic stripe that is a part of the badge. The encoding of data onto the magnetic stripe is based upon the American National Standard Magnetic-Stripe Encoding for Credit Cards, ANSI X4.16-1973.

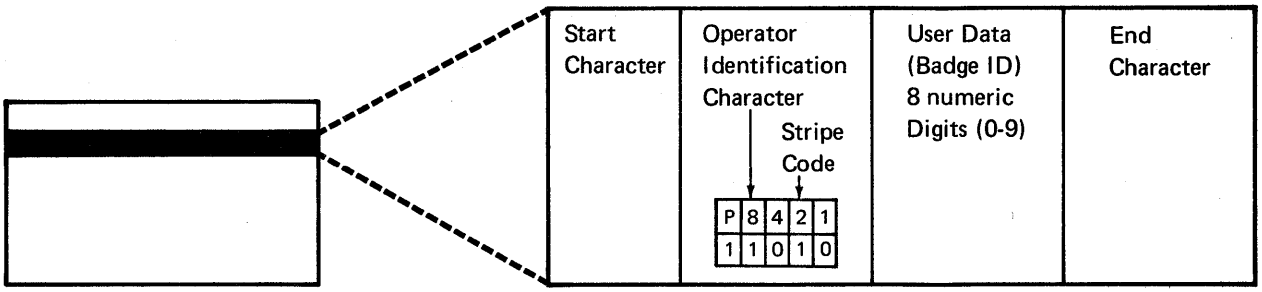
Using badge security requires that password security be on the system. The badge IDs are stored in the password security file.

The badge ID must be eight numeric digits (0-9).

Format of the Magnetic Stripe

The data encoded onto the magnetic stripe of your badge is read by a magnetic stripe reader.

The following diagram illustrates the general format of the magnetic stripe you need to have when using badge security.



Consult your local IBM branch office about the specific format used by a magnetic stripe reader.

MENU SECURITY

Menu security can be used to assign a menu and session library to a user and to optionally restrict that user to operating only from the assigned menu. The master security officer can assign or change menus for himself and for any other user. A security officer can assign or change menus only for system operators, subconsole operators, and for work station operators.

When a menu is assigned, it can be designated as mandatory or not mandatory. When the menu is mandatory, the operator can enter only a menu item number, an OFF control command, or an MSG control command after signing on. If the user attempts to specify a different menu on the sign-on display, the system does not allow the operator to sign on. When the menu is mandatory, the operator can use the System/34 HELP function to display information, but he cannot use the HELP function to run procedures. When the assigned menu is not mandatory, the menu is a default menu, which appears after the operator signs on but does not restrict the operator to using it.

Assigning a mandatory menu does not restrict the operator to using only that one menu or to a limited number of menu items. The mandatory menu can chain to other menus, and procedures run from a menu can display other menus by using the MENU OCL statement.

If a library name and a mandatory menu are assigned, the system does not allow the operator to sign on if he attempts to specify a different menu or library on the sign-on display. If a library name and a menu that is not mandatory (or no menu) are assigned, the library becomes the session library after sign on. The library assignment can be overridden on the sign-on display.

CAUTION
If a library and a mandatory menu are assigned, the user cannot sign on if the menu or library does not exist on the system. If the menu and library are specified as defaults (the menu is not mandatory), and if the menu or library does not exist, the user can sign on by entering zeros in all positions of the menu and library fields on the sign-on display.

FILE AND LIBRARY SECURITY

File and library security prevents unauthorized use of files and libraries and can be used only if password security is active. File and library security uses a file called the resource security file to store information about each protected file or library. The resource security file contains a record for each protected file and library. Each record contains the user IDs of authorized users of the file or library. For each authorized user, the record contains a code that identifies the user category.

For a file, the user's authority can be any of the following:

- The user is an owner and can:
  - Give file access to others
  - Rename the file
  - Read, display, add, update, or delete information in the file
  - Delete the file

- The user can read or display the information in the file, but cannot change the contents of the file.

- The user can read, display, and change the contents of the file.

For a library, the user's authority can be any of the following:

- The user is an owner and can:
  - Give library access to others
  - Rename the library
  - Display members in the library
  - Execute members in the library
  - Copy members from the library
  - Update members in the library
  - Delete members from the library

- The user can copy, display, execute, and change the contents of the library.

- The user can copy, display, and execute members in the library, but cannot update the contents of the library.

- The user can only execute members in the library. The user cannot be prevented from executing members in #LIBRARY; however, the ability to copy, display, or change members in #LIBRARY can be controlled.

  Note: If a mandatory menu is assigned, the user can execute only members that are executed from that menu or chained menus. Therefore, a user can be prevented from executing members in #LIBRARY if the assigned menu and chained menus do not execute #LIBRARY members.

A public access level can be assigned for a file or library. For example, the public access level for a library can be assigned as *execute only*. In that case, all system users can execute members from that library. However, only the users specified in the access list can read or change the member. No user can be restricted to a level of access lower than the public access level for the file or library.

An entry for a file or library can be placed in the resource security file before the file or library is created. When an attempt is made to create a new file or library for which an entry exists in the resource security file and resource security is active, the system checks whether the operator is authorized to change the contents of the file or library; the system creates the file or library and sets a flag in the disk VTOC (volume table of contents) entry indicating that the file or library is protected. If the operator is not authorized to change the file or library, the system displays an error message at both the requesting display station and at the system console; the display station operator must cancel the job step or the job.

When an attempt is made to use a protected file or library, the system ensures that the operator at the display station is authorized to use the file or library. If an attempt is made to do an operation that the operator is not authorized to do, the system displays an error message at both the requesting display station and at the system console. The display station operator must cancel the job step or the job.

The system makes special checks before allowing a display station to attach to an MRT program. The system checks whether the operator at the display station is authorized to execute programs from the active user library at the time the MRT was originally requested. If the MRT program allocates disk files, the system checks whether the operator is authorized to use those files. If the operator is not authorized to use the library or one or more of the files, the system displays an error message and does not allow the display station to attach to the MRT program. If the MRT program later requests a file to which the display station operator does not have authorized access, the security function displays a message at the system console, and the system operator must cancel the job step or the job.

Note: Operators of display stations attached to MRJE need not be authorized for all files and libraries used by MRJE. Operators must be cleared for only those files and libraries they use.

If a program attempts to acquire a display station, the system checks whether the operator at the display station is authorized to use the library that contains the program and whether the operator is authorized to use any files already allocated by the program. If the operator is not authorized to use the library or one or more of the data files, the system returns an error code to the program and does not allow the program to acquire the display station. If the program later attempts to use a file to which an operator at an acquired display station does not have authorized access, the security function displays an error message at the requesting display station; the operator must cancel the job step or the job.

SECURITY FILE LISTING

You can obtain printouts of the security file in four different formats. These different security formats are:

- System security options by user ID
- Resource security by resource name
- Resource security by owner ID
- Resource security by user ID

You obtain the printed listing of the security file either by using the PRLIST procedure or by running $PRLT with the appropriate OCL statements. When you run the security file listing procedure, a temporary file is created and used to help in processing the security records. The data in this temporary file is erased when the security file report is generated on the output printer you specify. For more information on the security file listing, refer to the PRLIST procedure in the *SSP Reference Manual*.

Notes:

1. Certain functions of the PRLIST procedure can be requested only by a master security officer.
2. Any listing that has passwords shown should be kept in a secure place to avoid disclosure to unauthorized persons.
3. The output from the PRLIST procedure exists as an entry of the spool file until it is printed. This output can be copied by either the system operator or the subconsole operator using the COPYPRT procedure. Multiple copies of the PRLIST output can also be made by the subconsole or system operator.

Interactive Communications Feature (SSP-ICF)

The SSP-ICF support on the System/34 allows programs on the System/34 to establish communications with a remote system and to communicate with a program or programs at the remote system. SSP-ICF also allows programs on a System/34 to communicate interactively with other programs on the same System/34.

SSP-ICF includes a common application program interface that allows access to data management support for specific communications subsystems. The following remote system communications interfaces and protocols are specifically supported by configured subsystems types:

- SNA
  - System/370 IMS/VS (using SLU type-P protocols)
  - System/370 CICS/VS (as a 3790 full function logical unit)
  - System/370 user-written communication support using SNA profiles TSP 3 or 4 and FMP 3 or 4
  - System/34 with SSP-ICF point-to-point
  - System/34 with SSP-ICF multipoint
  - 3601 Finance Communications Controller
  - X.21 Public Data Network (Japan and Scandanavian countries)

  Note: SNA communications to System/370 is supported through NCP/VS and VTAM or ACF/NCP/VS and ACF/VTAM.

- BSC (point-to-point or the System/34 as a multipoint tributary)
  - System/370 IMS/VS via IRSS (multipoint only)
  - System/370 CICS/VS (as a System/3)
  - System/3 Model 15 CCP
  - System/34 using SSP-ICF or batch BSC support (point-to-point only)

In addition, communications with the following devices and systems is supported via BSC protocols:

- 3741 Models 2 and 4 (point-to-point only)
- 3747 (point-to-point only)
- System/7 with MSP/7 (as a System/3)
- System/32 (point-to-point only)
- System/3 with MLMP or RPG TP
- System/38 using BSC support
- Series 1 (as a System/3)
- OS, OS/VS, DOS, or DOS/VS BTAM
- OS or OS/VS TCAM
- 5231 Model 2 (as a 3741 in transmit mode only)
- 3750 (World Trade only)
- 3705 using NCP EP or PEP
- 5110 (as a 3741)
- 5260 (as a 3741)
- 5280 (as a 3741)

In addition, Assembler language users running batch BSC jobs can send or receive variable length records, blocked or unblocked.

Also provided is a subsystem that allows a System/34 to coexist on a BSC multipoint line with 3270s. The host can be a System/360, System/370, or a System/3. The user can write programs that use the current set of host programs for 3270s. This support does not allow attaching 3270s to the System/34.

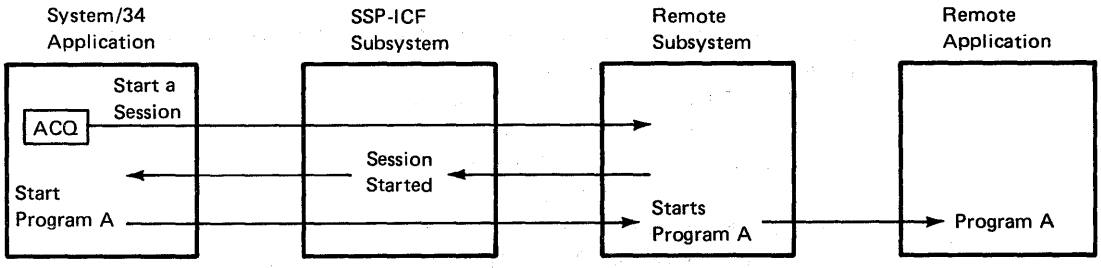
SSP-ICF also provides the Intra subsystem, which enables concurrently executing programs on the same System/34 to communicate with each other. No data communications line is involved.

SSP-ICF SESSIONS

A key SSP-ICF concept is that of a session. By definition, a session is the logical connection (or pipeline) between a System/34 application program and a remote subsystem. Thinking in terms of display station interaction with programs might help you understand the session concept.

Locally Initiated Sessions

As described earlier under *Program Attributes*, an application program can acquire a System/34 display station. That display station remains attached to the program until the program ends or until it releases the display station. In exactly the same way, a System/34 application program can acquire a session. The program itself is coded as if it were acquiring a display station for use as a WORKSTN file. The only difference is that instead of using a WORKSTN statement to define the requested display station, you use a SESSION OCL statement to define the requested session. The SESSION statement identifies the session ID used within the program and the location ID that was assigned during SSP-ICF configuration.

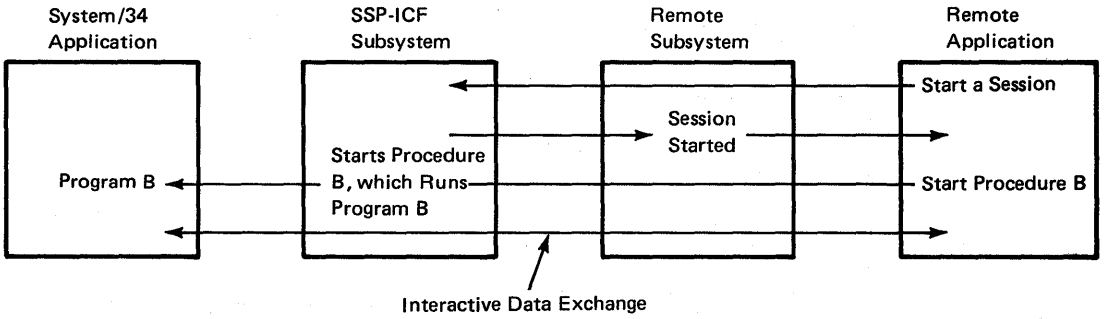


A session ends when one of the following events occur:

- The program requests that the session be ended (via a release or *end-of-session* operation). If a release operation is used to end the session, the session will end only if the release operation is successful.
- The program terminates.
- An error causes abnormal termination of the session.
- A normal disconnect occurs on a switched line.

Remotely Initiated Sessions

As described earlier, under *Program Attributes*, a requesting display station is automatically attached to an SRT or an MRT program when the program is initiated. When a procedure is evoked from a display station, that display station is attached to each SRT and MRT program that is run during execution of that procedure. Analogously, a program on another system for which a session was configured and enabled can issue a procedure start request for the System/34. The System/34 will execute the procedure and the requesting session will be attached to each SRT or MRT program that is run as part of that procedure. The requesting program can then communicate interactively with the program run on System/34.



The session ends when one of the following events occurs:

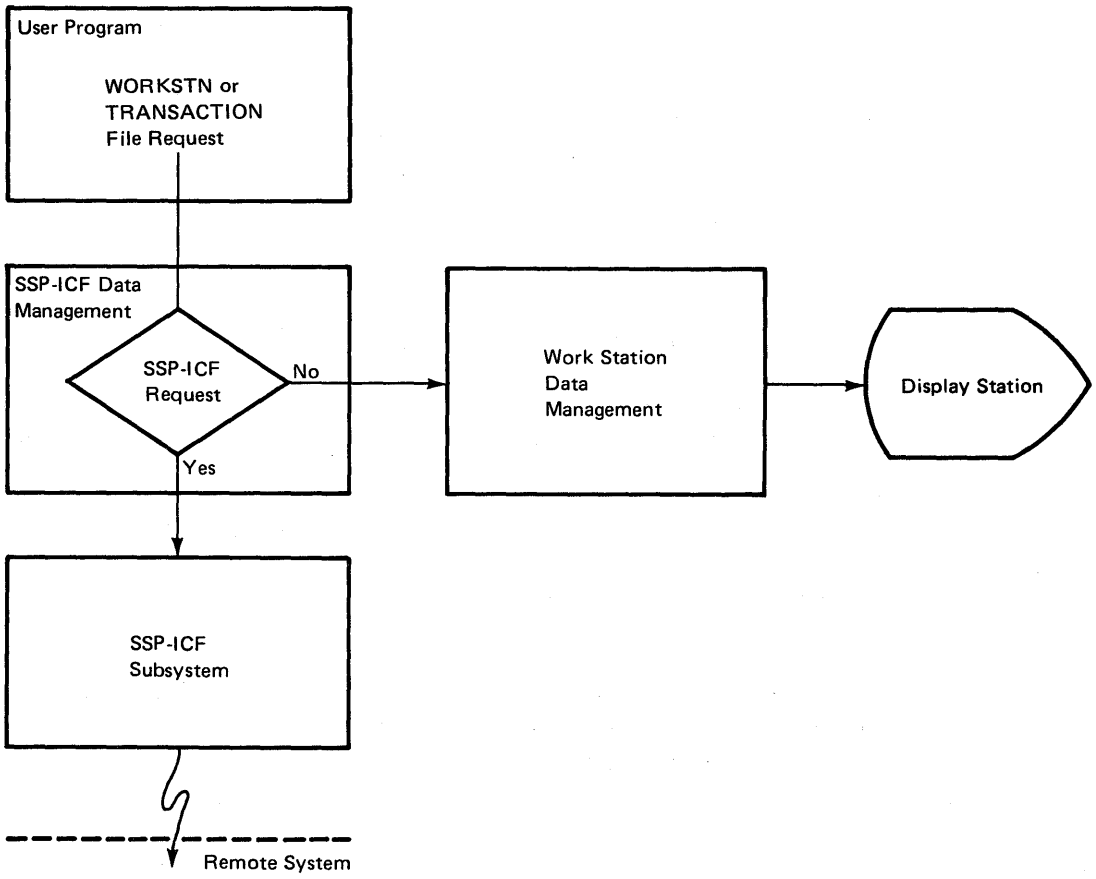
- A program issues an *end-of-session* operation.
- A program issues a *put-end-of-transaction* operation. If *end of transaction* is received, the program must issue an *end-of-session* operation to terminate the session.
- An error causes the session to be abnormally terminated.

When the session ends, all subsequent procedure steps, if any, execute with no requesting display station attached.

Note: Some ICF subsystems, such as BSC 3270, do not support remotely initiated sessions. Refer to the ICF reference manual to determine if the subsystem you are using supports remotely initiated sessions.

SSP-ICF DATA MANAGEMENT

RPG II programs use the WORKSTN file and COBOL programs use the TRANSACTION file to interface with SSP-ICF support. Basic assembler programs use assembler macroinstructions to interface with the SSP-ICF support. BASIC programs use the OPEN, CLOSE, READ, and WRITE statements to interface with SSP-ICF support. When a program issues an operation for the file, SSP-ICF data management processes the request. (SSP-ICF data management runs in the main storage nucleus area as a subroutine of the requesting task.) If the requested operation is for an SSP-ICF subsystem, SSP-ICF data management transforms the request into the format required by the communications subsystem. If the request is for a display station operation, SSP-ICF data management passes control to work station data management.



SSP-ICF data management and the communications subsystems use an area of nonswappable main storage as an intermediate buffer pool for sending or receiving data for SSP-ICF sessions. This buffer pool allows efficient overlapping of user program operations with the communications required to send or receive data. For output operations, SSP-ICF transforms the user request to the format required by the communications subsystem. SSP-ICF data management transforms the request by assigning and moving data from the user program to the nonswappable buffer space. The communications subsystem is informed of the request via a task post, and, if the operation does not require waiting for successful completion, SSP-ICF data management returns control to the user program. For input operations, the user program issues an accept input operation, which causes SSP-ICF data management to wait for data from either a display station or an SSP-ICF session. If data is received for an SSP-ICF session, the data is moved from the nonswappable main storage buffer space to the buffer in the user program.

Autocall Capabilities

The IBM System/34 autocall feature allows you to make calls automatically over switched MLCA lines without operator intervention. With the autocall feature, you will be able to do the following:

- Define multiple lists of phone numbers that can be accessed by the following communications subsystems:
  - BSCEL
  - BSC CICS/VS
  - CCP
  - SNA PEER
  - SNA 3270 emulation
  - SNA upline facility
- Execute both MRJE and SRJE tasks from a procedure from any work station, and call a phone number without system operator intervention
- Have repeated executions of both batch BSC or SSP-ICF job steps that call multiple locations
- Acquire SSP-ICF sessions and calls placed to the location without system operator intervention

To use the System/34 autocall support, do the following:

- Specify which lines are to be autocall lines when your service representative configures the microcode
- Create object members containing the phone numbers of the remote systems you wish to call
- Start the autocall task by running either a batch BSC job, an MRJE or SRJE utility, or an SSP-ICF job

For more information on how to use the autocall feature, refer to the *Interactive Communications Feature Reference Manual* if you plan to use autocall with SSP-ICF. If you plan to use the autocall feature with either MRJE or SRJE or batch BSC, refer to the *Data Communications Reference Manual.*

System/34 Finance Support Subsystem

The System/34 Finance Support subsystem allows application programs using the interactive communications feature (ICF) to communicate with all models of either the 3601 Finance Communication Controller or the 3694 Document Processor. A communications adapter for SDLC communications is required to use the finance support subsystem.

For more information about the finance support subsystem, refer to the *Interactive Communications Feature Reference Manual.*

DATA COMMUNICATIONS AND THE X.21 INTERFACE

The System/34 supports features that provide an interface to public data networks supporting digital communications. This interface is known as the X.21 interface.

To use the X.21 feature, your system needs a multiline communications adapter (MLCA). Each MLCA communications line can be defined as either switched or leased when using digital communications. The line speeds available are:

- 2400 bits per second
- 4800 bits per second
- 9600 bits per second
- 48 000 bits per second

The X.21 interface supports up to three switched or four leased lines. If one line is configured as a switched X.21 line, the other two lines may be configured as either X.21 or non-X.21 switched or leased lines.

The following table lists the communications support available with the X.21 interface:

Communications Component	Leased Lines	Switched Lines	
		Auto Call	Auto Answer
Remote work station support	Yes	Yes	No
Batch BSC support	Yes	Yes	Yes
MRJE and SRJE	Yes	Yes	Yes
SSP-ICF/BSC subsystems	–	–	–
BSCEL	Yes	Yes	Yes
CCP	Yes	Yes	Yes
CICS	Yes	Yes	Yes
IMS/IRSS	Yes	No	No
3270 Device Emulation	Yes	No	No
SSP-ICF/SDLC subsystems			
SNA upline	Yes	Yes	Yes
PEER	Yes	Yes	Yes
3270 SNA emulation	Yes	Yes	Yes
Finance subsystem	Yes	No	Yes

Communications Support Available with the X.21 Interface

Leased Lines

The leased lines must be defined as X.21 communications when your customer engineer does microcode configuration.

Switched Lines

For switched lines you must do the following:

1. Define which lines are to be X.21 lines during the microcode configuration.

2. Start the X.21 interface by running one of the previously listed communications components.

X.21 Autocall Feature

Using the X.21 autocall feature requires you to create object members containing the phone numbers of the remote systems. You create these object members by using the DEFINX21 procedure.

The DEFINX21 procedure is described in the *SSP Reference Manual.*

For more information on using the X.21 feature refer to either:

- *Interactive Communications Feature Reference Manual.*
- *Data Communications Reference Manual.*

Sample Inquiry Applications Using SSP-ICF

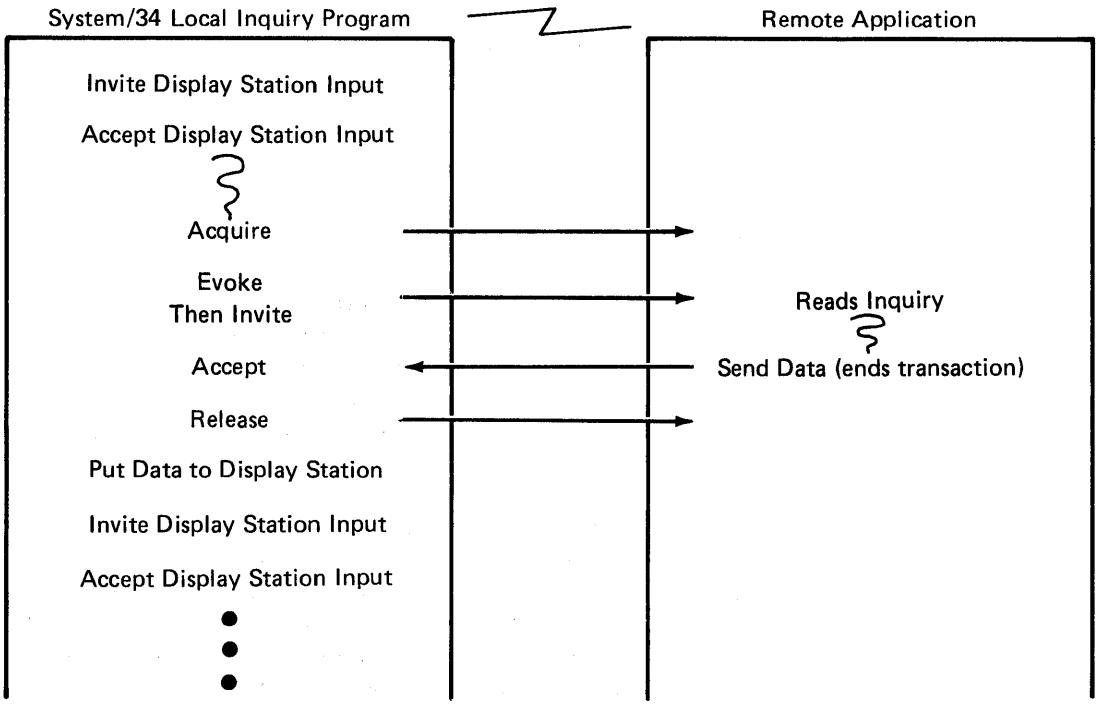
The following two examples show simple inquiry applications that use SSP-ICF.

Local Inquiry Program

In this example, a local inquiry program evokes a remote program to answer an inquiry. The local program could be an MRT-NEP that receives an inquiry request from a display station and, if necessary, evokes a remote program through an interactive communications session. After receiving the inquiry response from the remote program, the local program releases the session and displays the answer to the display station operator.

If the local program is an MRT-NEP, it remains active even if no display stations or sessions are attached. The program waits for a display station to request the program and make an inquiry. After receiving the inquiry, the local program determines whether the information is available on the local system or if a remote application must be evoked. If the information must come from a remote application, then the local program performs the steps described in the following paragraph.

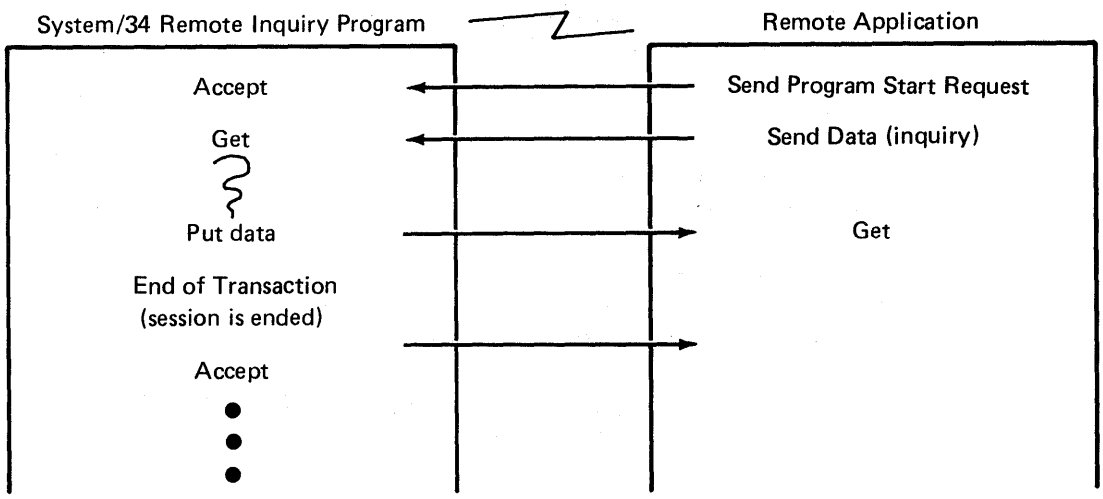
First, the program acquires a session corresponding to one described on a SESSION OCL statement. For information about acquiring a session, refer to *SSP-ICF Sessions* in Chapter 2. After successfully acquiring the session, the local application evokes the remote application. The evoke operation should be an evoke (with data), then invite. The evoke portion of the operation activates the remote program and includes the inquiry as data; the invite portion allows the remote application to send the results. Multiple accepts or gets are necessary if the response is expected to be multiple records. After receiving the response, the local program releases the session and displays the inquiry results to the display station operator. The program can then wait for the next inquiry. For information about SSP-ICF operations, refer to the *Interactive Communications Feature Reference Manual.*


System/34 Local Inquiry Program
Remote Application
Invite Display Station Input
Accept Display Station Input
Acquire
Evoke
Then Invite
Reads Inquiry
Accept
Send Data (ends transaction)
Release
Put Data to Display Station
Invite Display Station Input
Accept Display Station Input

Remote Inquiry Program

A remote inquiry program is started by a remote system to answer an inquiry. After being started, this program receives an inquiry from the remote system, sends the response back, and terminates (or handles other inquiries if it is an MRT program). The remote inquiry program could be the counterpart to the local inquiry program discussed previously.

The first operation should be an accept to receive the session ID and any data transmitted with the request, possibly followed by a get to receive the inquiry. The program must then perform any processing necessary to answer the inquiry before issuing one or more put operations to transmit the response to the remote system. The session ended when the application program successfully sent end of transaction. If the program is an MRT-NEP, it can issue an accept to wait for the next requestor.



Checkpoint Facility (For COBOL and Assembler Programs and Subroutines)

When a batch job will run for an extended period, you can provide for the periodic recording of information during the run. The checkpoint facility saves the status of the system, and preserves the associated resources used by the job step. Thus, if the program prematurely terminates, the checkpoint facility provides a means of restarting the program at an intermediate point. For more information about using checkpoints in a program, refer to the *Basic Assembler and Macro Processor Reference Manual* and the *COBOL Reference Manual*. If you intend to use the checkpoint facility for an interactive (nonbatch) job, refer to the restrictions and considerations for both checkpoint and restart to ensure that the program can be restarted. The restrictions and considerations are described later in this section.

When you use the checkpoint facility, a checkpoint record file is created for each job step. The checkpoint record file exists until the job step terminates normally. If a malfunction occurs, the checkpoint record file remains on disk until the program is restarted and terminates normally. The file labels for each checkpoint record file must be different from any other file on the disk. The file label is specified in the program. For information about determining the size of the checkpoint record file, refer to the *Installation and Modification Reference Manual*.

Files and libraries are marked as protected (checkpoint active) when they are being used by a program saving checkpoints. If a malfunction occurs, these resources remain protected just as if the program were still running.

When a checkpointed job step has run successfully, the checkpoint information is removed. The resources assigned to that job step become available for other tasks.

Checkpoint Restrictions

The following is a list of restrictions for a job step that requests a checkpoint:

- Disk files that are specified for the job step cannot use the DISP-SHR option in the FILE statement.
- All files, including offline multivolume files, must be allocated and then opened before any checkpoints are requested.
- There is enough space on the fixed disk for the checkpoint file.
- Checkpoint is not permitted if the program is using batch BSC communication lines; however, programs can be using remote work stations and SSP-ICF sessions.

If the requestor for the program is an SSP-ICF session, checkpoint is allowed if (1) the program was evoked with *end of transaction* and (2) either PDATA-NO (the default) was specified for the procedure that loaded the program or the program has accepted the data sent when the program was evoked.

The following is a list of restrictions for other users when another job step requests a checkpoint (the CATALOG procedure can be used to display the checkpointed status of the file):

- The total number of disk VTOC entries available for permanent and temporary files is reduced while checkpoint active files reside on disk. The amount of reduction is the sum of the following values:
  - The number of scratch and job files that are not allocated from the reserve area.
  - The checkpoint record file.
  - The reserve area file.
- Other jobs and system functions cannot access the checkpoint active files.
- Checkpoint active files may not be used as input for $COPY functions such as saving the file on diskettes or printing its contents.
- Checkpoint active files may not be deleted by $DELET or renamed by $RENAM.
- Librarian functions that require dedicated use of a library are not permitted. Therefore, a condense operation of a checkpointed library, including #LIBRARY, cannot be requested if the checkpointed program failed and has not yet been restarted.

Checkpoint Considerations

If you use the checkpoint facility, you should be aware of the following programming considerations:

- If an assembler program requests a checkpoint, the first checkpoint should be issued as soon as possible so that:
  - A valid checkpoint record exists.
  - The space for the checkpoint record file has been allocated.

- Each time an assembler program requests a checkpoint, the program should check the completion code in the parameter list.

- Make sure that operator instructions are complete, so that if a malfunction occurs, the operator knows the proper action to take. This action should include:
  - How to respond to checkpoint/restart system messages
  - When to run the CRESTART procedure

- The checkpoint record file must be a new file and cannot have the same label as an existing file. A FILE OCL statement must not be supplied for the checkpoint record file.

- Files may be modified by additions, deletions, and updates. The checkpoint function keeps track of the last record in the file when the checkpoint occurred. The restart function deletes all records added after the last checkpoint. Deletions and updates remain in effect unless the program has some means to restore them. The *COBOL Reference Manual* further describes considerations for checkpointed programs that update a disk data file.

- If a 2 option was selected for a message and then the 0 option was selected for message SYS-1314, the remaining job steps will run immediately. If you restart the program, those steps will run again after the checkpointed job step completes normally. If you do not want to execute the remaining job steps when the 0 option is selected, you can use a // IF ?CD?/3721 expression to branch around the remaining steps.

- The RESERVE statement is permitted for scratch and job data files. If this statement is used, the SSP will attempt to allocate these files from the reserve area.

- If jobs that take checkpoints terminate abnormally, all S (scratch) and J (job) files that are not in a reserve area have VTOC entries. If a restart is not done, these files are flagged as checkpoint active.

Restart Facility

The restart facility is a means of resuming the execution of a program from a checkpoint. Any operator may restart a checkpointed job step. When a checkpointed job step is restarted, the restart facility attempts to recreate the status of the job step as it was at the checkpoint.

The restart facility rebuilds and updates the system control blocks with the data saved in the checkpoint record. All the resources that the job step used are restored to the same status as at the checkpoint. This involves locating and allocating disk files, instructing the operator to replace a diskette if offline multivolume processing is used, restoring printer file status, and restoring the work station work area. The job step is then restored from the last checkpoint.

The restart facility sets the appropriate completion code in the checkpoint/restart parameter list. The assembler programmer should check for this completion code so as to determine whether the program is returning from a checkpoint or a restart. Return code checking enables the assembler programmer to perform any further recovery operation that the particular application may require. For further information about the checkpoint/restart parameter list, refer to the *Basic Assembler and Macro Processor Reference Manual.*

When using the restart facility, it is the user's responsibility to acquire any display stations that the job step requires, and to restore the proper displays and data to those display stations.

Restart Considerations

The following list describes restart considerations:

- The job step should be restarted as soon as possible after the malfunction because the checkpointed resources are not available for other jobs to use.

- When a checkpointed job step fails, the operator can remove the checkpointed job from the system if the job step is not going to be restarted. Until the job step is restarted or canceled, the job and scratch files associated with the job step will exist along with their VTOC entries; also, any checkpointed libraries including #LIBRARY, will appear as if they had a user. The CATALOG procedure can be used to display the status of checkpointed files and libraries.

- The local data area and external indicators (UPSI switches) are restored even if the restart function is run from a display station other than the one from which the program was checkpointed. Therefore, you should ensure that any jobs following the restart do not depend upon data that was in the local data area or external switches before the restart function was run.

- If resource security is active, the operator who runs the restart function must be on the list of users for the user files and libraries for the job step being restarted, as well as all job steps that follow the checkpointed job step.

- A RESERVE OCL statement must not be used before the restart function is run.

- MRT programs are restarted as SRT programs.

- If a checkpointed program has not yet been restarted, do not run the RELOAD procedure. A restart is not possible if the RELOAD procedure has removed message members, load members, procedures, or menus from the system library or from a user library.

- If a checkpointed program has not yet been restarted, do not delete or rename a checkpointed user library.

- Acquired display station sessions or acquired SSP-ICF sessions that were active when a checkpoint occurred must be acquired again by the program at restart time even if REQD-YES is specified on the WORKSTN OCL statement. The program must keep track of the display screen formats and data that were displayed at the time of the checkpoint and must restore the displays and data.

- The diskette files for offline multivolume files are not marked as protected by the checkpoint function. If a failure occurs, the operator should ensure that the diskette files are not used as input or output to any other program until the failing program has been restarted.

- If a checkpointed program is restarted and goes to normal completion or if the checkpointed program was executed as part of a procedure and the procedure completes normally, the following conditions occur:
  - If a menu was active at the time the program or procedure was initiated, that *checkpointed menu* is restored even if a different menu was active when the restart function was run.
  - The session library, menu library, and message member libraries that were active when the program or procedure was initiated are restored even if different libraries were active when the restart function was run.

Printed Output

When spooling is not active during a checkpointed job step, all records up to the latest checkpoint, including the ones following it to the point of the system failure, are printed. When the job step is restarted, the printed records requested after the last checkpoint are printed again. This can produce an overlap in the output.

If print spooling is active, overlapping of printed output usually does not occur. There is an overlap in the printed output only when DEFER-NO is specified in the PRINTER statement even though print spooling is active. When spooling is active, the spooled output generated after the program is restarted is placed in a different entry in the spool file. Therefore, output from before the malfunction and from after the restart are in two separate spool file entries.

Note: If a checkpointed program fails, the spool file should not be reformatted before the restart function is run. If the spool file is reformatted and the program is restarted, spooled output (the output generated from the start of the program until the last checkpoint before the failure) is lost.

Nonrestartable Job Step

It is possible that certain conditions will prevent $RSTRT from restarting the job step. The job step might be nonrestartable when:

- A permanent disk or diskette I/O error occurred.
- A failure occurred before the first checkpoint was complete.
- A user library was deleted.
- The RELOAD procedure was run.

Removing Checkpointed Jobs

If a job step is nonrestartable or if the operator does not wish to restart the job, the system operator has the option of removing the checkpointed job from the system. The following describes three ways in which a checkpointed job step can be removed from the system:

- If the operator inquires out of or cancels a checkpointed job or responds with a 2 or 3 option to any message, a SYS-1314 message is issued. By responding with a 1 option to the SYS-1314 message, a specific checkpointed job is canceled.
- By using the DELETE parameter in the CRESTART procedure, the operator can cancel the checkpointed job step. For example, the checkpoint record file for a job step was labeled CK1. To cancel the checkpointed job step, the operator can enter:

  ```
  CRESTART CK1,DELETE.
  ```

- During file rebuild, the operator is prompted about whether to remove all checkpoint active files from the system. If the operator selects the Y (yes) option, all checkpoint active job steps are removed from the system.

Operator Considerations

When working with checkpointed jobs, the system operator and display station operator should be aware of the following considerations:

- An SRT must be restarted from a work station. If the display station does not have the same logical ID as the original requestor, the following may occur:
  - I/O to a specific display station will probably fail.
  - History file continuity will be lost.
  - ?WS? OCL substitution will cause unpredictable results.

  Note: An SRT program cannot be restarted by the EVOKE operation, by the // EVOKE OCL statement, or from the input job queue. Other types of programs can be restarted in those ways. For example, a program that was run from the input job queue was checkpointed and the checkpoint record file was called CKP1. To restart the program from the input job queue, the operator can enter:

  ```
  JOBQ ,CRESTART,CKP1
  ```

- If the configuration has been modified for date format, belt image, forms number, forms length, or translation table, a diagnostic will be issued to which the response can be to terminate the restart or to reinstate the previous values that were saved during the checkpoint. If the operator chooses to reinstate the previous values, the following statements apply:
  - The belt image remains changed until the next sign-on or until the SET procedure is used to change the belt image.
  - The translation table remains changed until the next sign-on or until the SET procedure is used to change the translation table.
  - The other values remain changed until the checkpointed job terminates. At that point, they will return to the values they had before the restart.

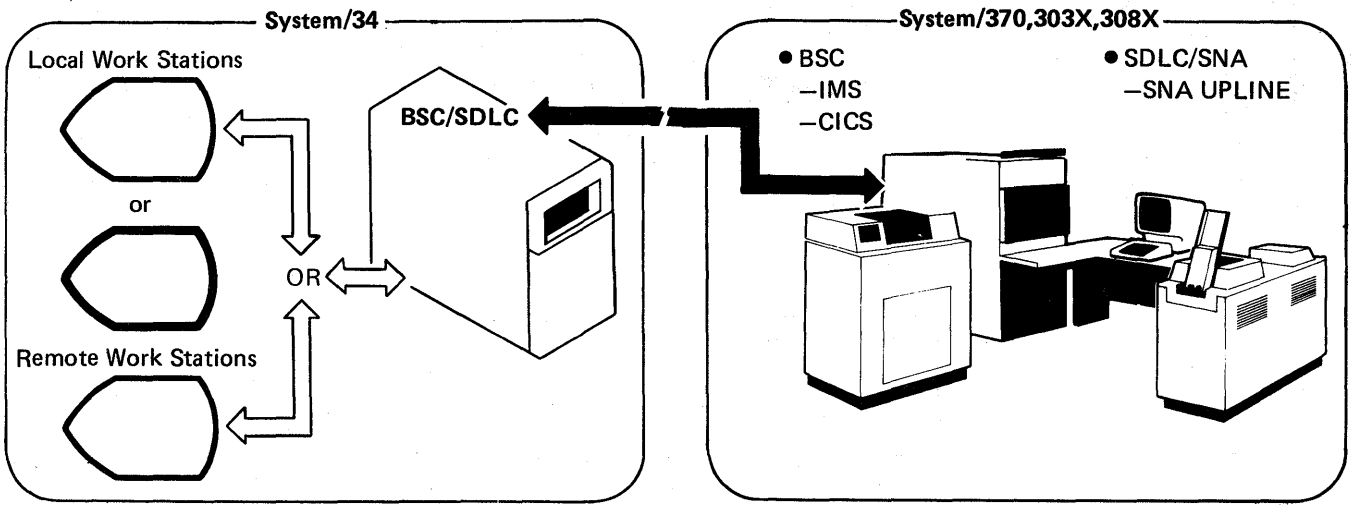
- The tasks priority will not be restored to the value at the checkpoint.

System/34 and Distributed Data Processing Environments

The System/34 has the data processing capabilities which allow you to use your System/34 in a distributed data processing environment. A distributed data processing environment is where the power of the computer is shared by many users who may be in different locations. These locations can be in another part of the main computer site or in a different city. You can use the System/34 as a processor terminal, a host system, a subhost system, or a peer connection. You can also process files that are located on a System/3 Model 15D or another System/34.

System/34 as a Processor Terminal

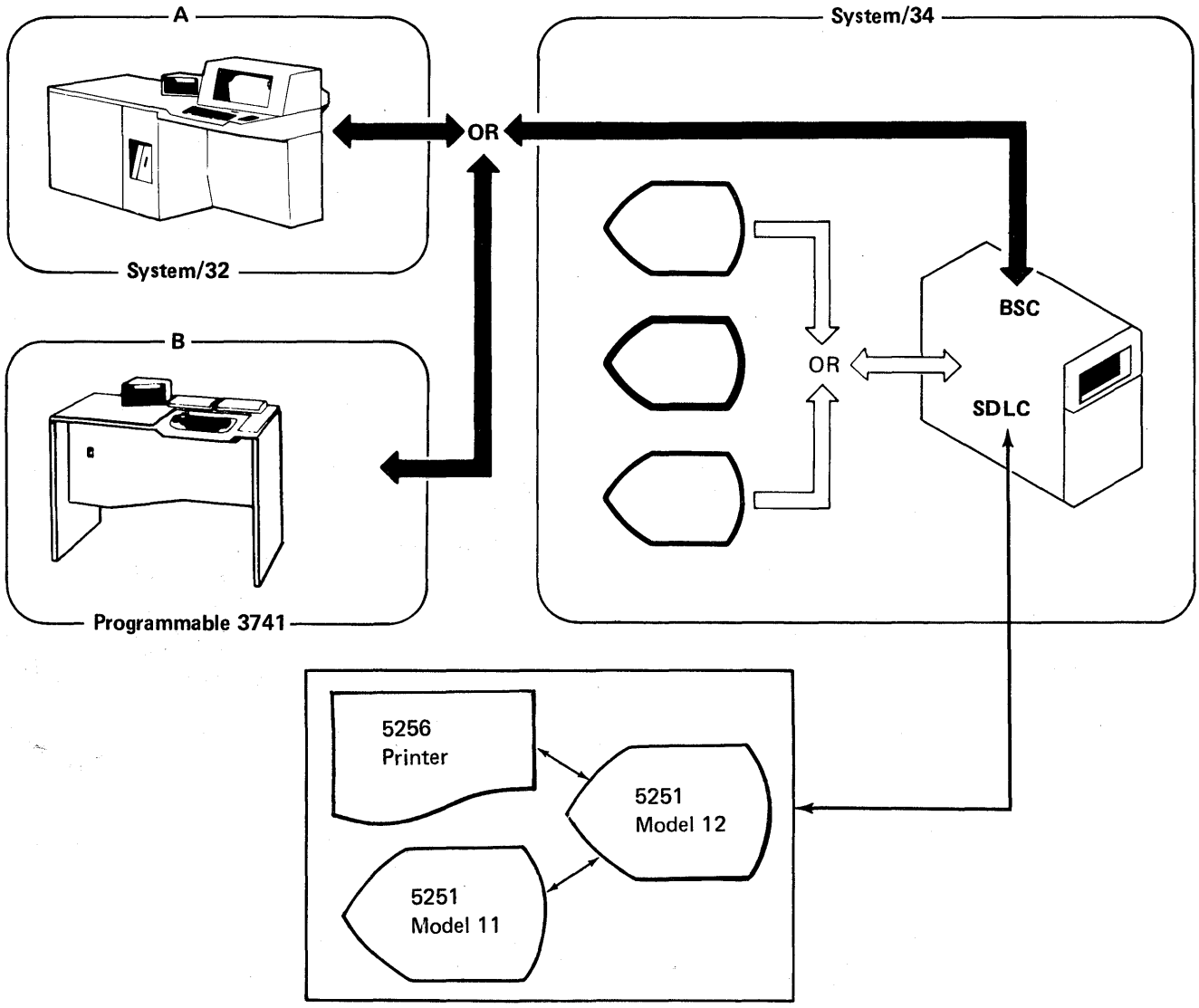
System/34 as a processor terminal is more applicable in firms with multiple locations where the facilities that are remote from the central location are large enough to require their own processing capability. In the following examples, the remote location has a System/34 and the central installation has a System/370 (the host system can be a System/34 or a larger system). Data transfer is over communications lines (switched or nonswitched) between the processors. Transmission is made to, and perhaps from, the central office's system to provide summary data as well as updated file information. In some cases, the remote processor may be used for remote job entry into a host system. An advantage of using System/34 is that it has multiprogramming, which allows communications programs to execute concurrently with other programs.



If the host is a System/370 or another System/34, communication can be via BSC or SDLC. If the host is any other type system, communication can be only via BSC.

System/34 as a Host System

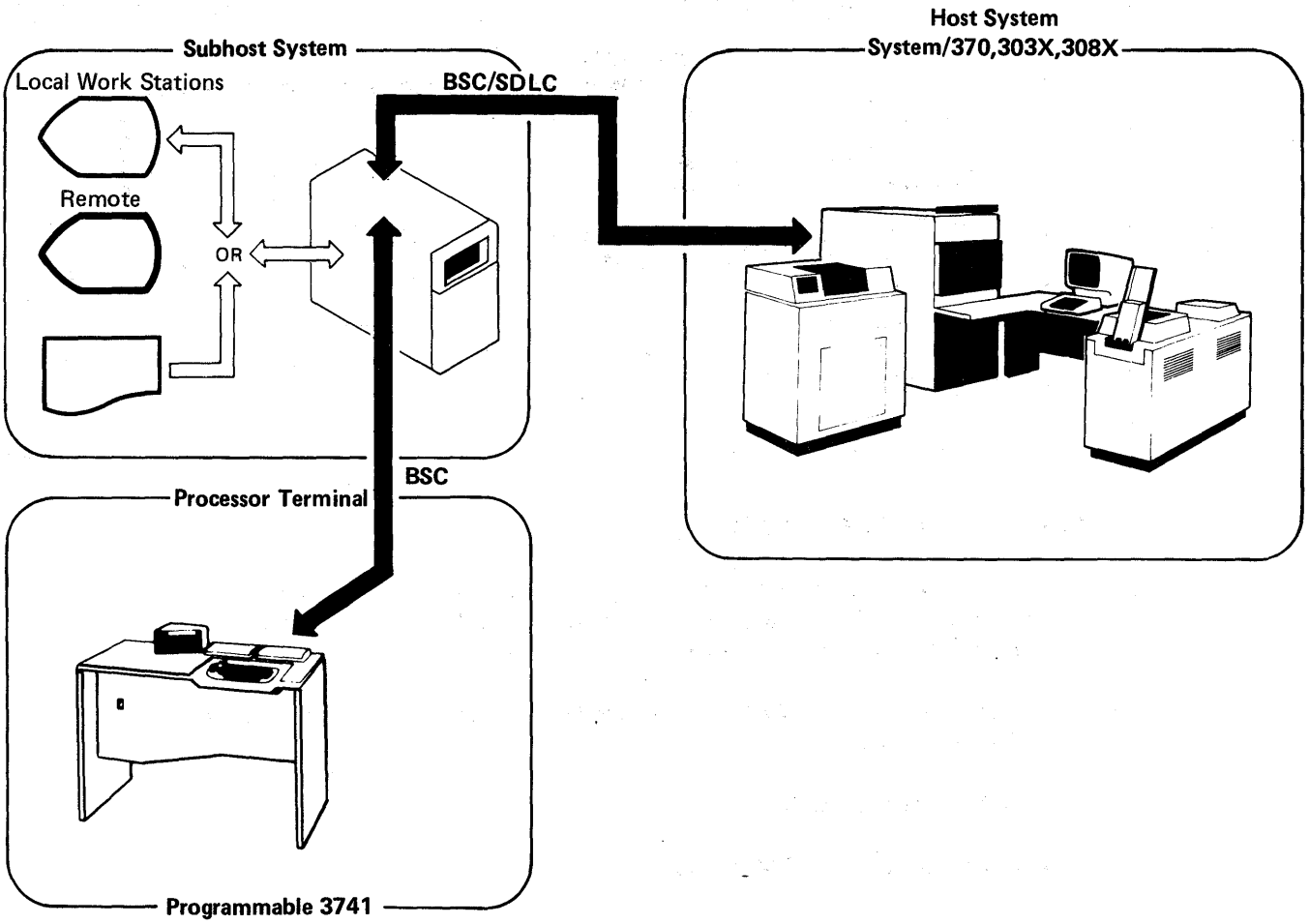
System/34 as a host system is applicable to many common business services. Many organizations have data that should be entered into a central control file. Some examples are attendance reporting, customer order masters, and inventory. Firms that have remote locations can use terminals to transmit or receive data using nonswitched or switched communications line connection to the host System/34. For example, a business has several remote locations. Location A uses a System/32 for payroll processing and location B uses a programmable 3741 to enter inventory transactions; the other locations use online 5250 devices. Summary data from locations A and B are transmitted to the System/34 using a switched communications line, while a nonswitched line is used to communicate with the 5250 devices.



Multiprogramming is an advantage of using System/34 as a host system.

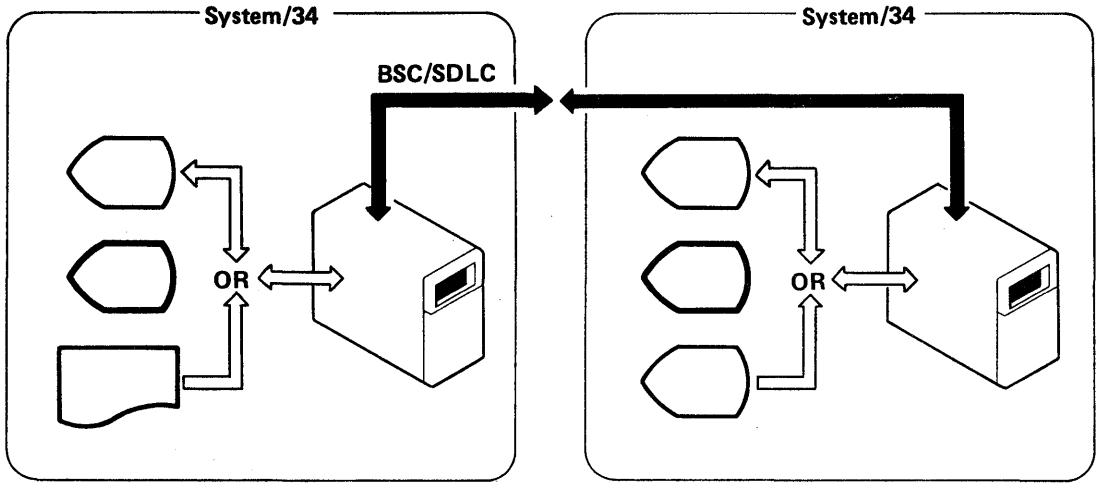
System/34 as a Subhost System

This category is a combination of the first two. The subhost serves as a host for some processor terminals, and also as a type of processor terminal for another host. A subhost system provides all the advantages of terminal entry to interact with centralized data files in a system and to communicate with the central office's system. The central (host) system has the main data files, and each subhost has a subset of those data files that can be accessed by the terminals attached to the subhost.



System/34 as a Peer Connection

System/34 as a peer connection is applicable to many common business services with one or more remote System/34s. The peer connection allows System/34 to utilize data on remote systems and to start remote procedures to offload processing. A peer connection provides System/34-to-System/34 processing without having to go through a host. This processing can be done on a point-to-point connection using BSC or SDLC, or on a multipoint connection using SDLC.



DISTRIBUTED DISK FILE FACILITY

The distributed disk file facility allows you to access data files stored on another System/34 or a System/3 Model 15D. You do not have to recompile your programs to use this feature.

The distributed disk file facility is available by PRPQ only. The following is the PRPQ number list:

- P84037 for System/34
- P84038 for System/3 Model 15D

For more information about the distributed disk file facility, refer to the *IBM System/34* and *System/3 Model 15D Distributed Disk File Facility Reference Manual*, SC21-7869.

Chapter 3. Design Considerations

During system design, you typically need to make decisions regarding the following items:

- Displays and menus that operators use
- Input documents and printer forms
- Files
- Applications and programs
- System security and integrity
- Documentation
- Remote display stations

For example, you might need to answer questions such as:

- What information should operators see on their displays and how can I format the information so that it is readable?
- Should I design similar forms for the various printers? If not, how should they differ?
- Which file organizations should I use and when should I use them?
- How should I design the records in the files?
- What design decisions affect response times?
- Which application should I design first?
- Which data entry programming method should I use?
- Which programs should be SRT programs? Which should be MRT programs? Which should be never-ending programs?

Design considerations for all these questions and more are presented in this chapter. As you read these considerations, keep in mind that no two businesses are likely to have the same design concerns. Therefore, the information in this chapter is necessarily general in order to apply to as many situations as possible.

Display Design

This section describes some key elements of display design. The guidelines presented, although representative of most display design, might not apply in all circumstances. The topics are offered more for your consideration during display design than for rules that you must follow.

Display design is concerned primarily with the way data is displayed to the operator and the way that the operator responds to this data. Generally, input displays should be designed for ease of data entry, and output displays should be designed for ease of reading. Many displays show output as well as accept input. These displays are the most challenging for the designer, who must make tradeoffs between ease of data entry and ease of reading. This section might help the designer make decisions about the tradeoffs.

System/34 operators might be miles from the computer room or at least far enough away so they cannot bring their questions or problems to this room. Therefore, displays that these operators use should be clear, complete, and concise.

Operators should feel that their display stations are helping them be more productive. You might help operators attain a positive attitude for using their display stations by involving them in the display design. For example, ask for opinions of some sample displays that you plan for them to use and ask how they might improve them.

Experienced operators might be able to use displays that have few operator instructions on them, and they might be more productive if the response times are short. Inexperienced operators might need more guidance from the displays, and they might tolerate longer response times.

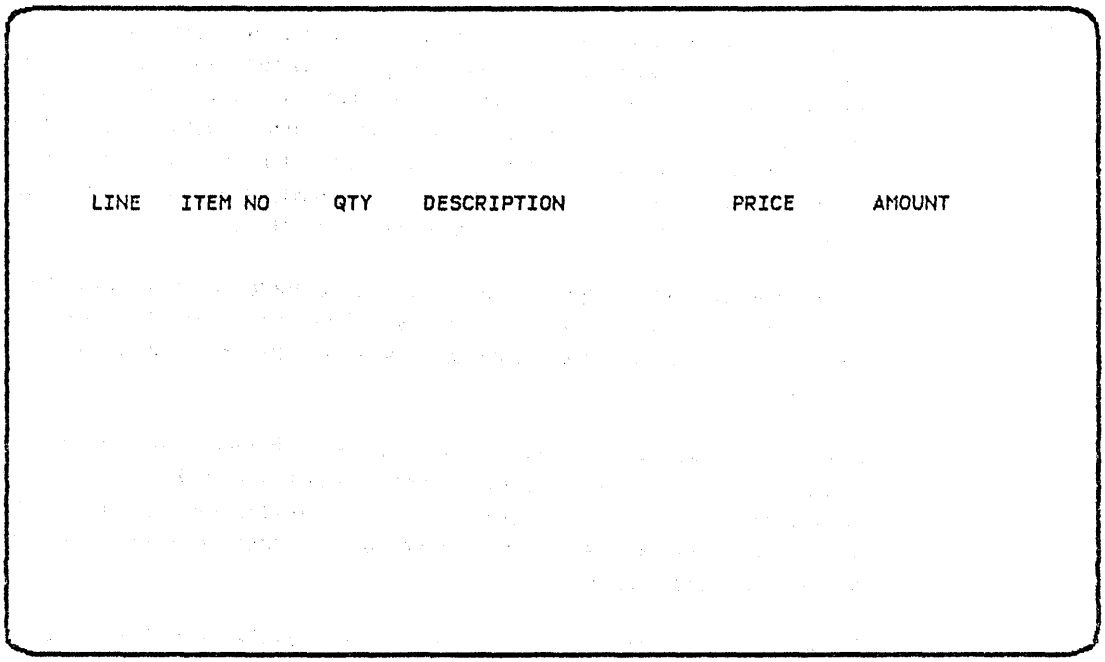
As you read the following display design guidelines, you might consider the amount of experience your display station operators have and plan your displays accordingly.

IDENTIFY THE DISPLAYS

Each display could be identified by a title or heading and a nondisplayed ID. The title should be as concise as possible, yet meaningful. Titles, which are usually centered on or near the top line of the display, could be intensified or underlined.

PROVIDE MEANINGFUL HEADINGS

Descriptive headings help operators understand what they see and do. For example, the headings on a display for entering line items from an order might be:



You might need to compromise between the number of descriptive headings you use, which makes a display easier to understand, and the length of the response time. Too many headings might degrade performance, especially on remote display stations. For those displays, you might want to minimize the number of characters on each display by abbreviating or eliminating the headings.

PLAN READABLE DISPLAYS

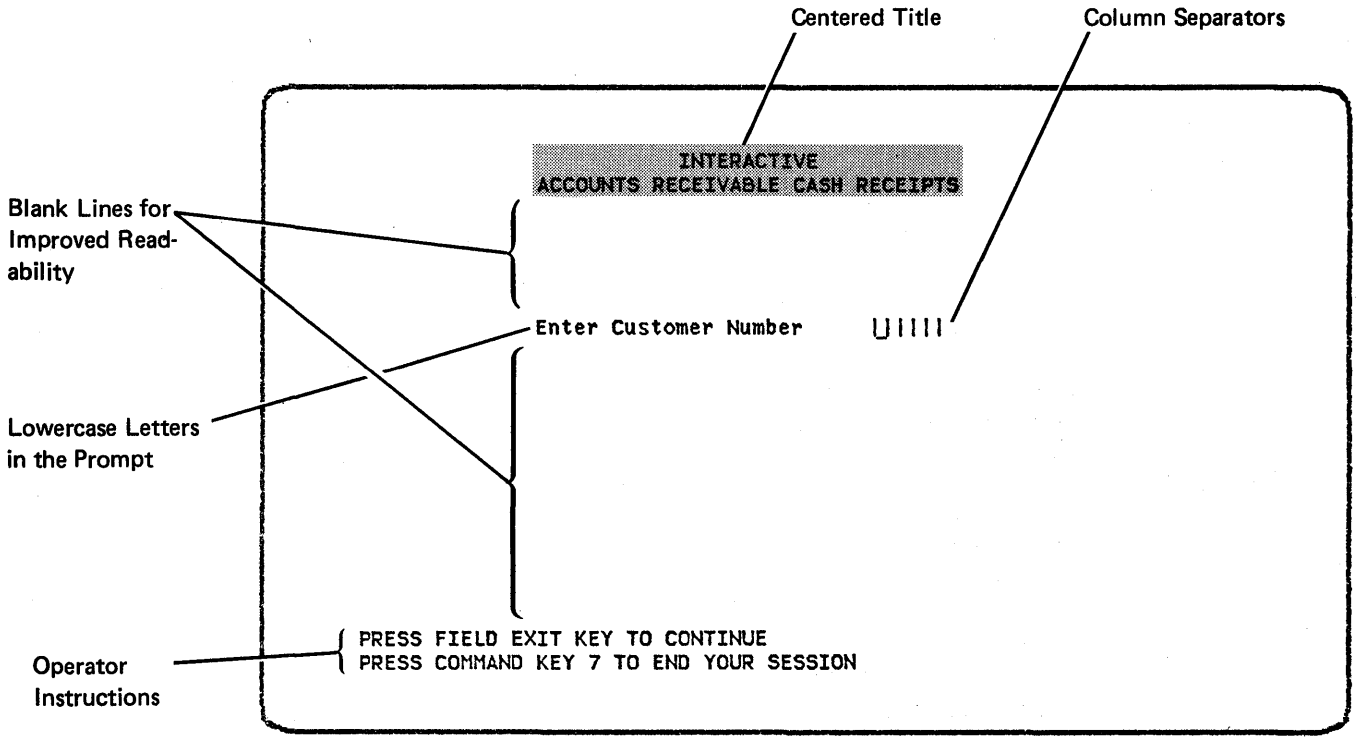
The basic rule of designing displays is to make the display screen easy to read. Display screens are the easiest to read when they are not cluttered with extra information. You should try not to put too much information on one screen.

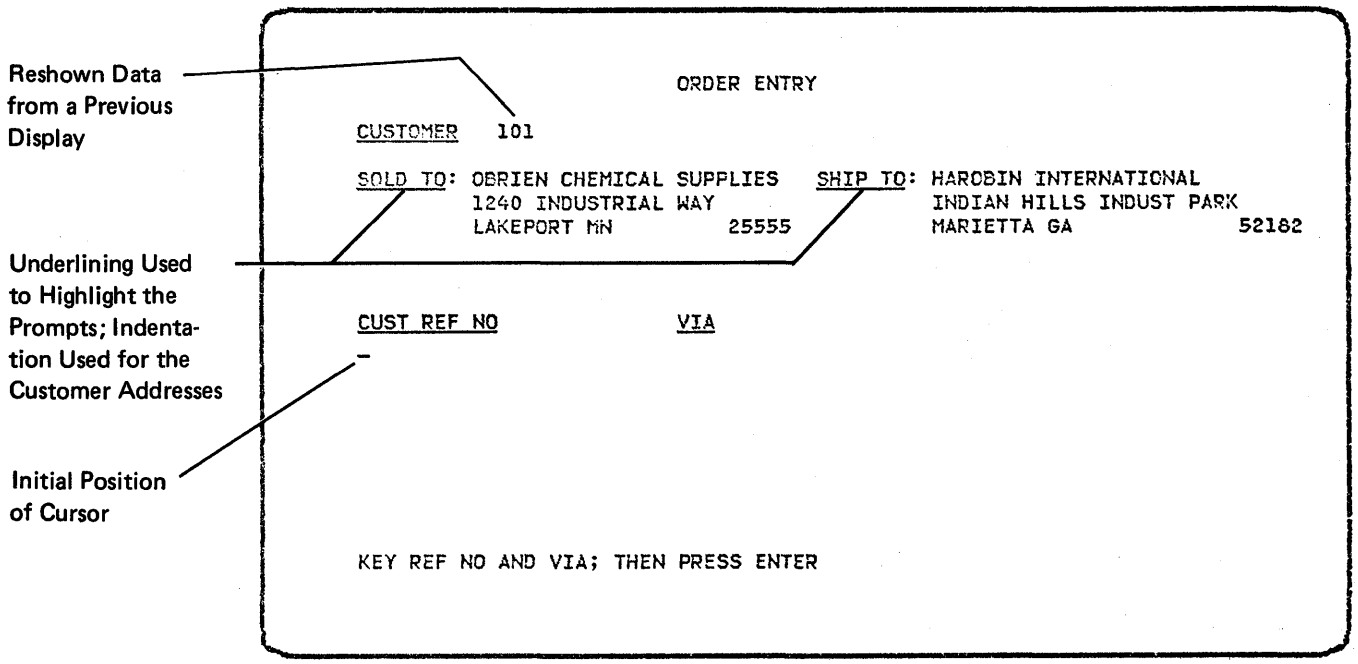
The following suggestions are provided to help you plan readable displays:

- Use blank space to separate data items. Blank space is the most effective, least cluttering, separator.
- Organize data in columns or lists. Text could be left-justified; numeric data could be right-justified and aligned on the units position.
- Present information in some recognizable order for ease of scanning. For example, put historical dates in chronological order.
- Use complete words rather than contractions.
- Use lowercase letters in prompts. (Lowercase letters do not print on a 5211 Printer or a 3262 Printer unless it has a 96-character print belt.)
- Use column separators in input fields.
- Highlight new, added, or referenced information when a display is reshown.
- Use blinking fields sparingly. Blinking should be used only for urgent, attention-getting purposes.
- Arrange fields so that the most frequently used fields are recorded first, followed by less frequently used fields.
- For inquiries, show only the expected data in a readable sequence.
- Avoiding unnecessary information such as asterisks that outline information, which reduces the amount of data sent from the program to the display station.

Avoiding unnecessary data is especially important for displays shown at remote display stations. Unnecessary data transmitted to a remote display station can cause lengthy response times.

The following displays show use of some of these design guidelines:





DISPLAY A SMALL AMOUNT AT ONE TIME

The displays should be kept uncluttered and include only meaningful information. For example, do not display the entire record on an inquiry if the operator normally looks at only one or two fields. Instead, display the necessary fields and, perhaps, provide a function that allows the operator to display the entire record when it is required.

Large displays are not intended only for displaying more data; part of the advantage of large displays is that they allow more formatting freedom. For example, double-spaced lines might make a display more readable.

Certain applications accumulate data from display to display. To a point, such an accumulation might be desirable. However, if you are not careful, the display can get too cluttered.

You might use nondisplay fields for information that is not needed by the operator but needed for your programs. For example, the display station ID or a screen format ID might be needed by a program, but it might not need to be seen by the operator.

MAINTAIN CONSISTENCIES AMONG DISPLAYS

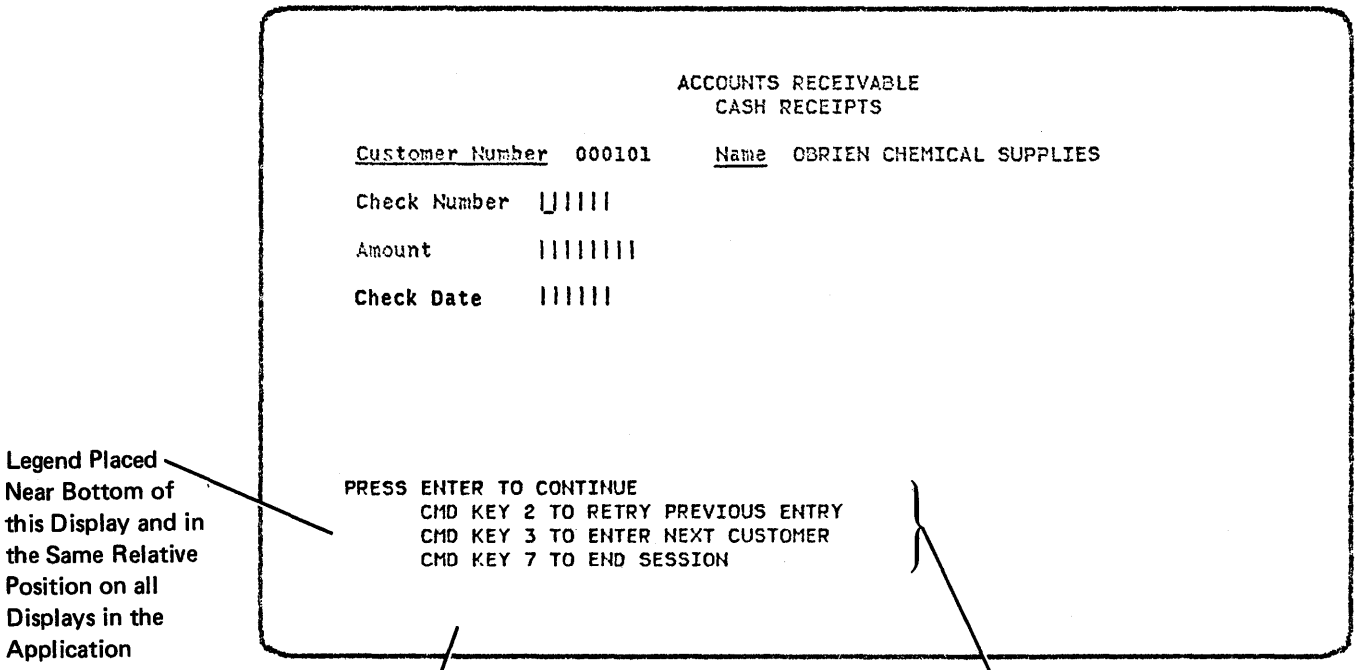
Each application has its own display screen requirements, but good design requires display consistencies among applications. For example, terminology, abbreviations, and codes should be consistent from one application to another. Consistency is particularly important when the same operator does more than one application. The standards established within an application are even more important than those between applications.

Use of keys should be standardized for displays and applications. For example, avoid allowing command key 7 to end a job on one display and command key 9 to end the job on another display. Of course, a key sometimes has to be used in an application-unique function, but a legend should tell the operator about the nonstandard use of the key.

Reserve areas of the display for certain types of information and maintain the areas in the same relative locations on all displays. For example, try to provide operator instructions and display messages near the bottom of the displays.

Finally, try to highlight a field consistently, whether via underlining, blinking, high intensity, or a reversed image.

The following display illustrates some of the consistency guidelines:



KEEP OPERATOR RESPONSES SHORT

Whenever possible, keep operator responses short but complete. These responses can include codes or abbreviations, but only if the operators are trained to use them.

Cursor positioning by operators should be minimized. Instead, displays should be designed so that operators need not frequently space over unused fields. The cursor might be positioned by the program to avoid this skipping over fields.

Consider blinking the cursor to draw the operator's attention to its initial position.

PROVIDE ONE IDEA FOR EACH DISPLAY

Whenever possible, a display should contain information concerning only a single aspect of an application. For example, one display should not be used to inquire into a file *and* perform an update at the same time. Concentrating on a single idea at a time decreases the possibility of an operator error. The following display is an example of allowing one function per display. The legend of command keys at the bottom of this display shows additional functions that the operator can select.

```
    30000    A & A GROCERY

A & A GROCERY                              PHONE:  312 / 555-1734
P O 15159                                  SALESMAN 16
E. DUNDEE        IL 60118


                          --- A/R BALANCE ---

   BALANCE FORWARD

PREVIOUS BALANCE            579.04       LAST PAID ON  9/02/78
   CHARGES                1,313.00
   PAYMENTS                 874.46       DETAIL FOR AMOUNT NOW DUE-
   ADJUSTMENTS                .00         CURRENT PERIOD          438.54
* AMOUNT NOW DUE          1,017.58         OVER 30 DAYS            579.04
   FUTURE CHARGES             .00         OVER 60 DAYS               .00
* TOTAL AMOUNT DUE        1,017.58         OVER 90 DAYS               .00
CREDIT LIMIT -            1,500.00         UPAID LATE CHGS            .00

COMMAND KEYS-
 1 RESUME SEARCH     2 NEW SEARCH       3 BILLING DATA      4 A/R BALANCE
 5 SALES DATA        6 PRICE INQUIRY   24 SIGN OFF
```

ACKNOWLEDGE OPERATOR INPUT

Operator interaction with a display station is usually conversational; for example, an operator makes an inquiry, and the display station shows the requested data. For this reason, you should be concerned about how long an operator waits for the System/34 to respond.

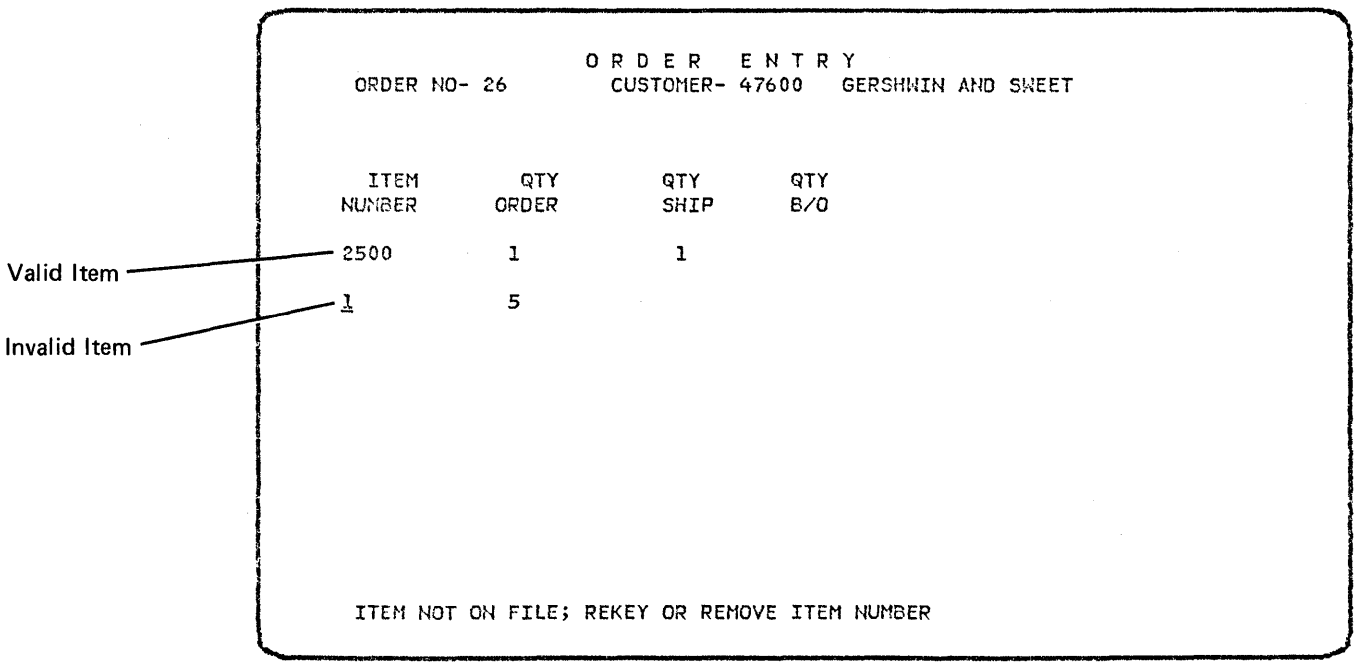
If a program takes a relatively long time to respond to an operator, you might want to display an in-process message immediately after the program receives the operator's input. For example, you might want to do this for a program that does extensive calculations with the operator's input. Acknowledging operator responses at remote display stations might affect their response times.

MAKE ERROR CORRECTION EASY

On System/34, the number of input errors might be reduced by detecting the errors as they occur and notifying the operator so that he can correct them. One way to design a display to inform an operator of an error is to use the lower part of the display for error messages.

You might reserve a fixed number of message positions on the next-to-last line of the display. The bottom line of the display is usually reserved for system error messages rather than your messages. The message field can be conditioned by an indicator so that the proper message is an output field when an error occurs. Indicators can be used to condition the attributes of the message and the field(s) in error, and indicators can also be used to position the cursor.

The following display shows the use of some of these guidelines:



PROVIDE A MEANS FOR HELP

Operators should know what actions to take when problems occur. Problem recovery steps should be included in the written operating procedures. In addition to the written steps, further information about the error should be available for the operator to display. This information might be requested via a command key or function key.

MAKE THE OPERATOR FEEL PRODUCTIVE

Because the operator is using the display station as a means to do a job, the display station should be easy to use and allow the operator to do a better job. Application programming and display design should not bore, scare, or annoy the operator. Try to use only as many features of the display station *as are necessary*. A display that has too many blinking fields or too much underlined data and highlighted information might only confuse and frustrate the operators.

Whenever possible try to give the well-trained operator the chance to take shortcuts from one display to another.

DOCUMENT THE DISPLAYS

Printed copies of your displays can be made by using the Print key. These copies can be labeled and stored with the $SFGR utility output for the displays. If possible, use a printer that can print lowercase letters. The 5256 and 5225 printers can print lowercase letters. The 5211 and 3262 printers can print lowercase letters only if they are equipped with a 96-character print belt. If lowercase characters cannot be printed, they can be changed to uppercase characters by using translation tables. Refer to the IMAGE OCL statement description in the *SSP Reference Manual* for details.

Make the Screen Look Like the Source Document

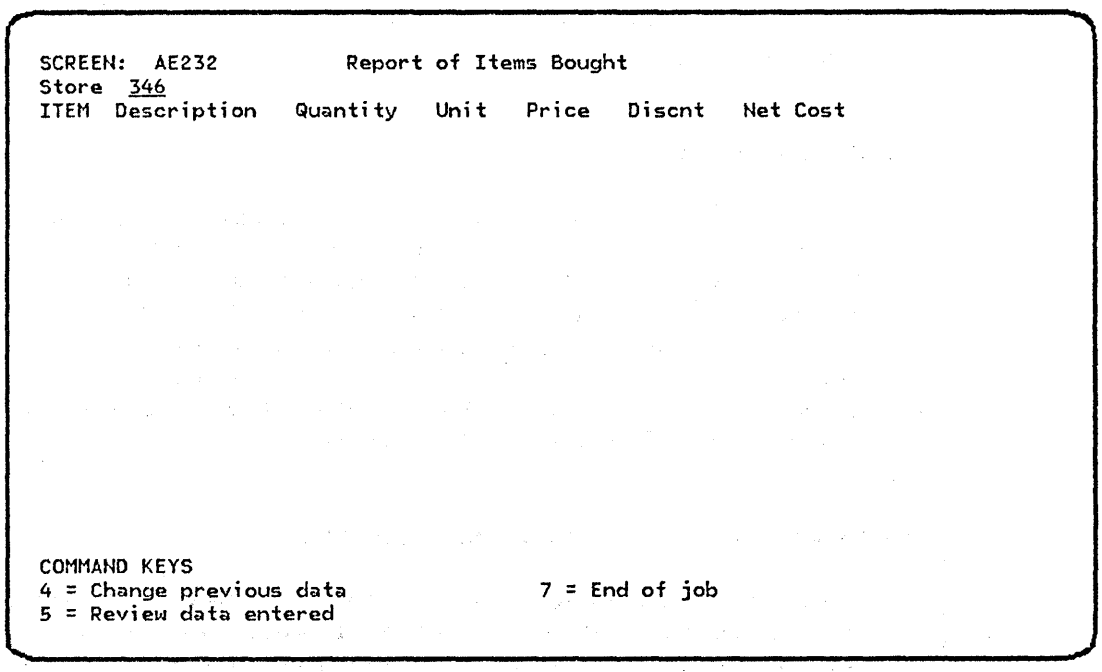
If the operators are entering data onto a screen from a particular document, try to design the screen display to look like the document. This technique is especially helpful if the operator's work includes a lot of data entry. The following display illustrates this principle.

Audio/Eyesight
1816 West Tuckey
Phoenix, Arizona 85015

Store Number 346

Daily Report of Items Bought

Item Description	Quantity Bought	Unit	Price	Discount	Net Cost
7653 Air Wrench	2	Doz	38.95	10%	70.11
3489 Widget	1	Doz	4.00	5%	3.80
4300 Balsam Wedges	150	Doz	12.00	10%	1620.00



Use SDA as a Documentation Aid

When creating or updating screen formats you can use SDA to help document your displays. The following example shows the documentation provided by SDA for a newly created screen format.

```
   SCREEN DESIGN AID                                  DATE  07/07/81     TIME  15.56

     1...+...10....+...20....+...30....+...40....+...50....+...60....+...70....+...80
 1 **                                                                            **  1
 2 **                                          ACME INC WEEKLY BILLING           **  2
 3 **     ACCOUNT IDENTIFICATION  21-______    STATUS SCREEN                     **  3
 4 **                                                                            **  4
 5 **                 CUSTOMER NAME    ________________                          **  5
 6 **                 STREET ADDRESS   ________________                          **  6
 7 **                 CITY _______________   STATE __     ZIP CODE ______       **  7
 8 **                                                                            **  8
 9 **                           INVOICE DATA SECTION                             **  9
 0 **                                                                            ** 10
 1 **   INVOICE NUMBER          INVOICE DATE        INVOICE AMOUNT     STATUS    ** 11
 2 **    A211302                 07/06/81           $142.30                      ** 12
 3 **    A411202                 07/07/81           $156.78                      ** 13
 4 **                                                                            ** 14
 5 **                                                                            ** 15
 6 **                                                                            ** 16
 7 **                                                                            ** 17
 8 **                                                                            ** 18
 9 **                                                                            ** 19
20 **                                                                            ** 20
21 **                                                                            ** 21
22 **                                                                            ** 22
23 **                                                                            ** 23
24 **CMD KEY 3 INVOICE STATUS       CMD KEY 7  RETURN TO BILLING PROCESS         ** 24
     1...+...10....+...20....+...30....+...40....+...50....+...60....+...70....+...80

          FORMAT.... WEEKLYAR                        SOURCE MEMBER NAME.... BILLING
                                                     LIBRARY............... DRFLIB
```

Menu Design

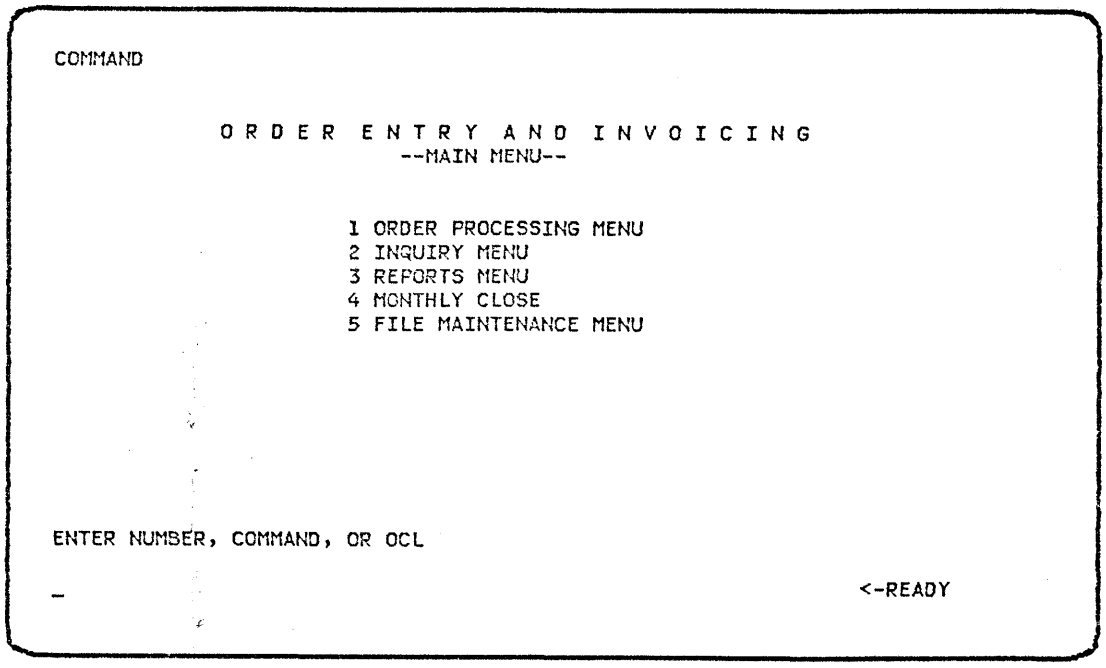
A menu is a displayed list of functions from which an operator can select an item. Using a menu, an operator enters a number that specifies what he wants to do. You might have the operators use menus so that they can start their jobs by selecting an item number rather than entering a command statement or OCL statements. Well-designed menus can shorten the time an operator takes to do his work and can reduce his chances for error. Your menu design might ensure that the operator does his jobs at the proper time in the proper order.

As described under *Menus* in Chapter 2, System/34 provides two menu formats: fixed and free-form. Fixed-format menus have two columns of pre-numbered items, and you can specify descriptions for as many of the numbered items as you want. On free-format menus, most of the lines (3 through 20) are available for you to format as you want.

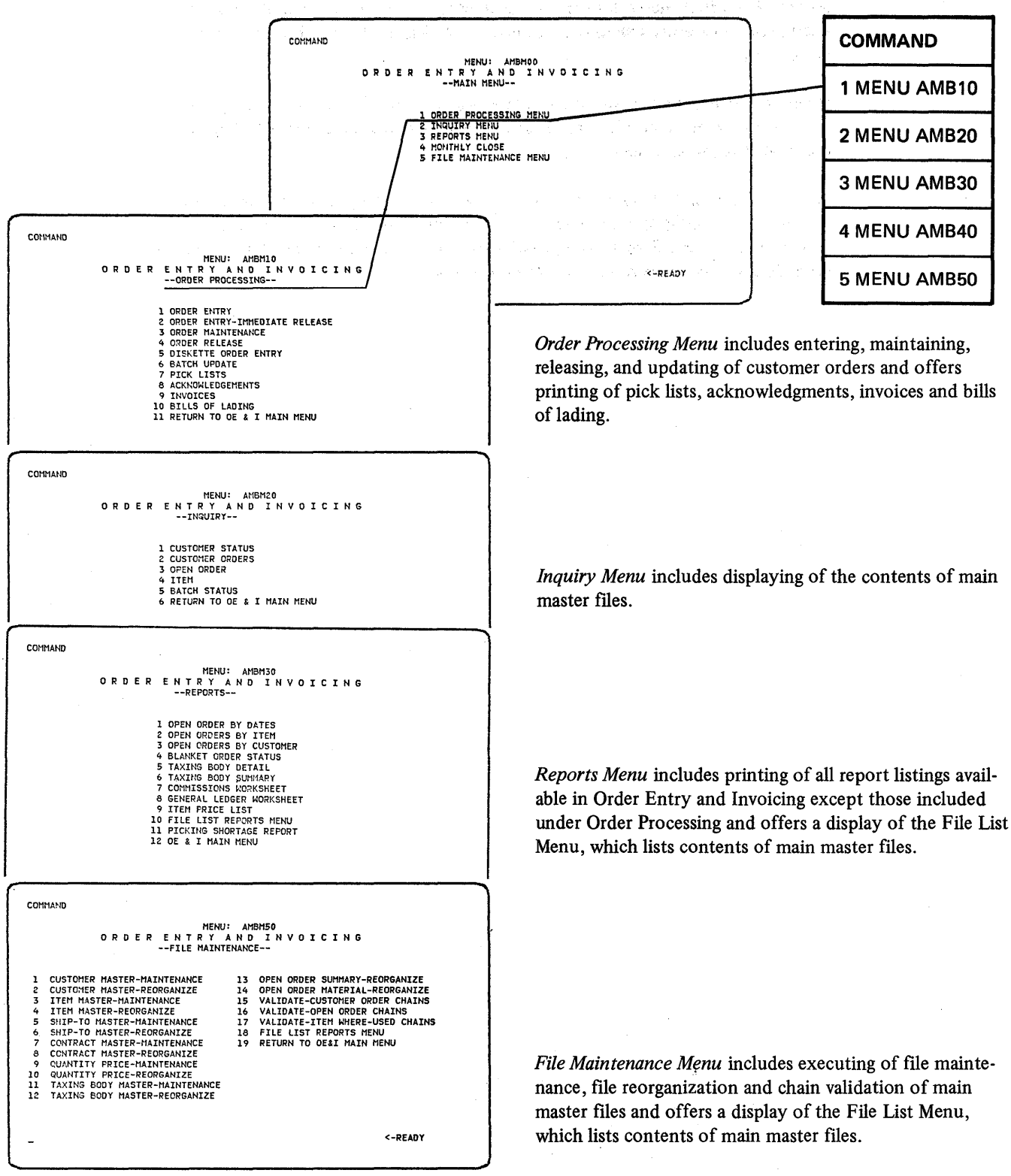
For either menu, you might consider placing the most frequently selected items near the top of the display so that the operator spends less time scanning for them.

Menu items should not be abbreviated unless you are sure that the abbreviations would not confuse the operators. Also, menu items should be meaningful. For example, Order Release is a more meaningful item than, ORDREL, the name of the program that releases orders.

Menu chaining, a good technique to use for applications on System/34, helps organize an operator's work by guiding him to the displays from which he does his jobs. This technique uses a main menu that categorizes the jobs that can be done. For example, the following menu is a main menu for an order entry and invoicing application. Notice that each item except Monthly Close causes a lower-level menu to appear.



The following figure shows some of these lower-level menus and the commands used to chain the menus.



Order Processing Menu includes entering, maintaining, releasing, and updating of customer orders and offers printing of pick lists, acknowledgments, invoices and bills of lading.

Inquiry Menu includes displaying of the contents of main master files.

Reports Menu includes printing of all report listings available in Order Entry and Invoicing except those included under Order Processing and offers a display of the File List Menu, which lists contents of main master files.

File Maintenance Menu includes executing of file maintenance, file reorganization and chain validation of main master files and offers a display of the File List Menu, which lists contents of main master files.

When an operator selects item 1 from the main menu, the MENU command is executed and the Order Processing Menu appears. When the operator selects an item from that menu, he sees either another lower-level menu if there are additional order entry categories to select or a display on which he can begin order entry.

When you chain menus, you might allow ways for operators to redisplay the main menu. Also, you might allow ways for experienced operators to bypass the menu chains and directly begin their jobs.

The // MENU OCL statement or the MENU control command are useful when you are constructing a menu chain. For more information about constructing menus, refer to the *Screen Design Aid Reference Manual* or to the BLDMENU procedure in the *System Support Reference Manual.*

Forms Design

Computer input and output forms should not be overlooked during system design because they are important interfaces between the system and the users of your system. These forms are possibly the only contact that some of the business' customers and employees have with the system; therefore, their design can influence impressions of the system and the business.

DESIGN CONSIDERATIONS FOR OUTPUT FORMS

System/34 supports the 5211 Printer, the 3262 Printer, the 5224 Printer, and the 5225 Printer which are line printers, and a 5256 Printer, which is a character printer. Each printer has unique forms design considerations.

On the 5256 Printer, printing is done one character at a time by a print head that must move to the appropriate position on the line. This head movement takes time. Therefore, you might consider designing your forms to reduce the amount of head movement required. For example, try not to center one or two fields on a line if the fields need not be centered. Left-adjusting them on the line might shorten the printing time. Placing fields in a horizontal line rather than spacing them vertically and minimizing the horizontal space between fields might also shorten the printing time. Planning horizontal lines so that fixed-length fields print first and variable-length fields print last allows the print head to advance to the next line as soon as the last character prints on the line. For example, an item description field might be a good variable-length field to print last on a line. Finally, using a large form and printing a small amount of information on it might be inefficient because many new line returns might be required to print the data. Therefore, keeping the forms as short as possible might improve printing speed.

Figures 3-1 and 3-2 illustrate an initial design and an improved design for an output form used by a 5256 Printer.

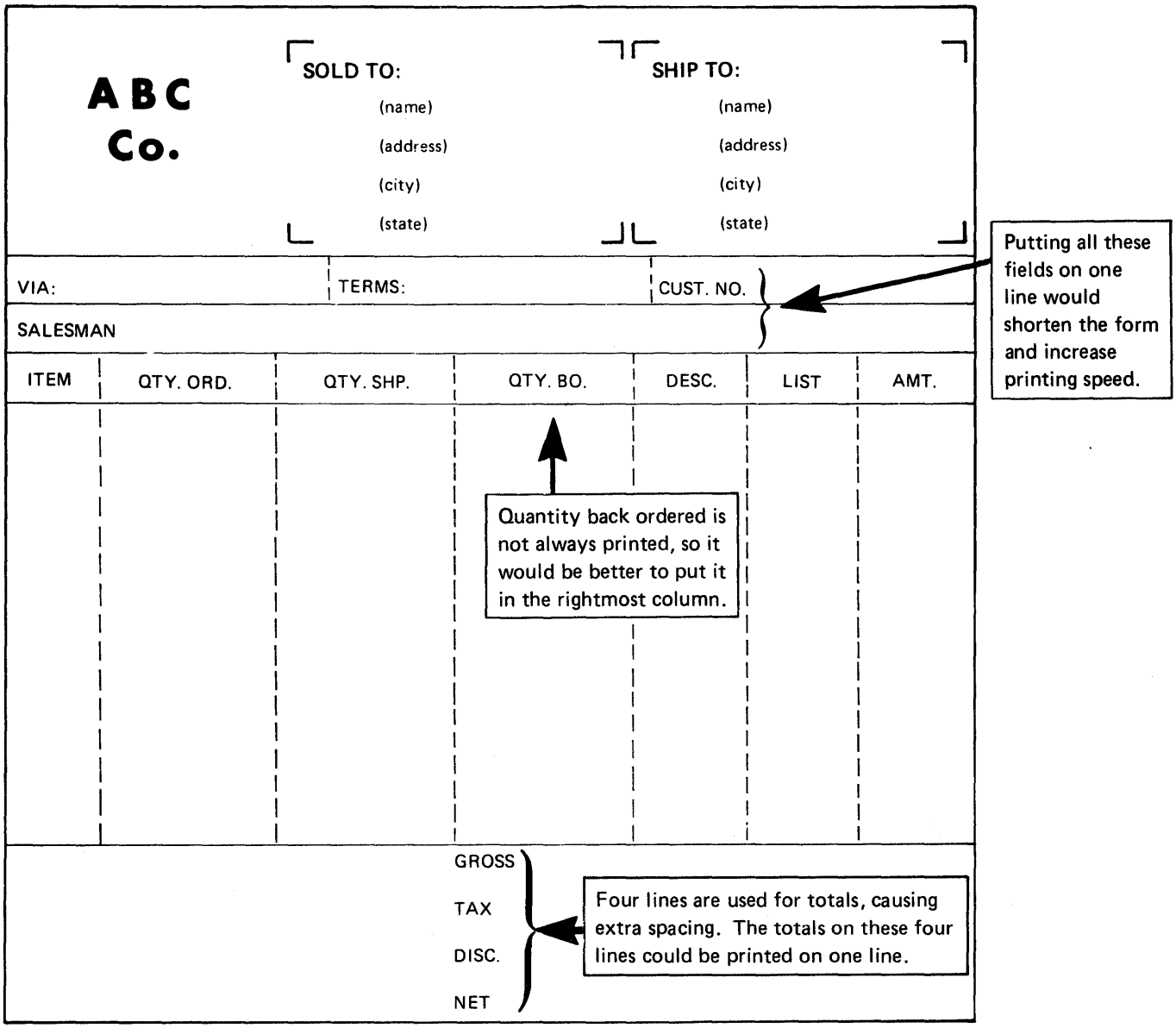


Figure 3-1. Initial Forms Design for a 5256 Printer

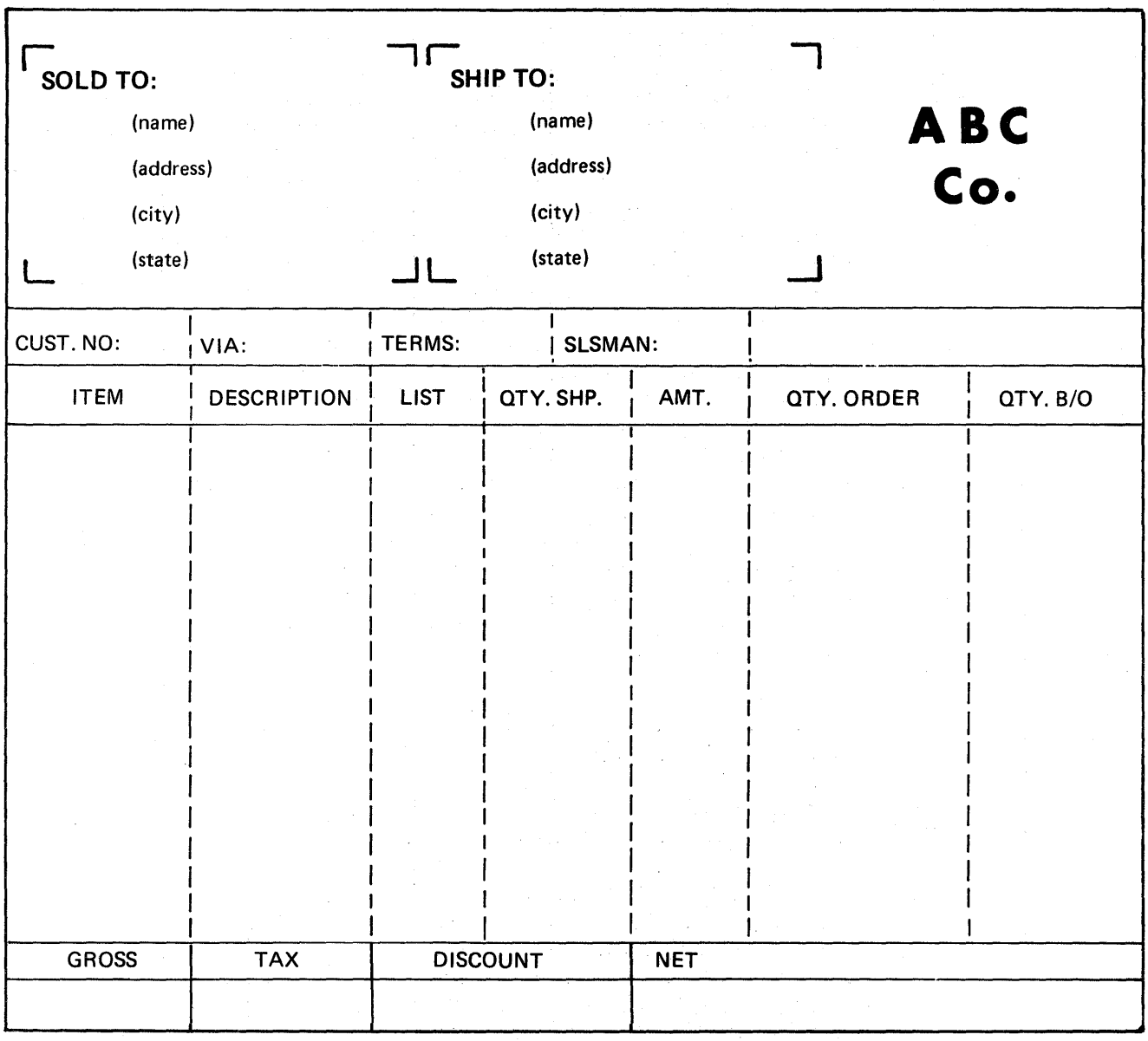


Figure 3-2. Improved Forms Design for a 5256 Printer

On the 5211 Printer or the 3262 Printer, one line is printed at a time by a character belt rather than by a print head moving from character to character across a page. Good design techniques for a line printer should therefore attempt to reduce the number of lines printed on each form. Increasing the number of characters printed per line might shorten the time required to print the form. An example of this would be to combine two lines into one print line. Consider using as wide a form as possible and as short a form as possible. Finally, try to space lines so that line skipping rather than line spacing can be used because line skipping is usually faster than line spacing.

For preprinted forms such as picking slips or invoices, consider shading alternative lines; this technique can make a long list of items more readable. Design and order your preprinted forms well in advance of when you plan to use them. Request a proof of the form so that you can verify its accuracy before it is printed.

Refer to the *SSP Reference Manual* for information about the number of printed lines allowed per page.

Refer to the following manuals for information about the physical dimensions of printer forms:

- *3262 Printer Component Description and Operator's Guide*
- *5211 Printer Component Description and Operator's Guide*
- *5224 Printer Models 1 and 2 Operator's Guide*
- *5225 Printer Operator's Guide*
- *5256 Printer Operator's Guide*

DESIGN CONSIDERATIONS FOR INPUT FORMS

Well-designed input forms might help shorten data entry time, reduce the number of data entry errors, and improve operator satisfaction with the system. Forms that will be used for data entry at a display station should match the input displays that operators use. For example, the input fields should be in the same order and have the same headings on the form and on the display. Usually, the form should be designed first and then the display could be designed to match the form.

To shorten the time required to fill out your forms, you might design them so that the data can be recorded in the order that it is usually received. You might consider placing mandatory entries first, followed by the optional entries, which might minimize skipping over fields. If you use special codes, consider listing them on the input form so that the operator does not have to look for their meanings.

File Design

One of the most important design activities is file design. This activity can significantly influence system performance, data security, data maintenance, data accessibility, and data recovery in case of a system failure. This section describes some considerations for attaining a good file design, particularly in the following areas:

- File organization
- Record design
- Record blocking
- Physical I/O and logical I/O
- Access algorithms for direct files

FILE ORGANIZATION

File organization can affect system throughput and display station response time in an interactive environment. Because much of the file processing is random, the choice of file organization is usually between indexed and direct. Indexed organization offers a wide variety of processing methods; however, direct file organization provides the following advantages that can contribute to an efficient system design:

- Fewer accesses to the disk can improve response times.
- File recovery in the event of a system failure is easier for direct files than for indexed files.
- Sharing files between programs is simplified when direct files are used.

These advantages can be critical to the performance of the system, especially for files that are frequently accessed. However, sequential and indexed files can be used in an interactive environment. Sequential organization is useful for files that are not processed randomly, such as some logging files. Indexed organization might be satisfactory for master files if the file is not too active and if short response times are not necessary.

The following information provides considerations for choosing the organization of your files. These considerations are *not* rules that you must follow, but guidelines that might help you organize each of your files so that you attain the best system performance possible.

Master File Organization

A master file is relatively permanent and is often used in several jobs with several other files. When you choose an organization for a master file, consider these processing requirements for it:

- What are the organizations of the other files that are processed against the master file? If the other files are ordered, which means that they are sorted in the same sequence as the master file, the master file could be processed consecutively and, therefore, a sequential or indexed organization for the master file would be most efficient.

  If the other files that you process against the master file are unordered, the master file needs to be indexed and processed randomly by key, or the master file needs to be direct. Processing an indexed file randomly requires a disk access to read the key and another disk access to read the record. Processing a direct file randomly requires one disk access per record unless the record has synonyms and is usually faster than processing an indexed file.

- How do other jobs process the file? If the master file is used in several jobs and its records are processed both in order and randomly, either an indexed or direct organization should be a better organization than a sequential organization.

- Does the master file require sorting? If so, consider that indexed and direct files can be sorted, but the sorted file is a sequential file. Rather than keeping the sorted file as the master file, you would need to keep the original unsorted file.

- Can operators inquire into the master file? If so, consider how necessary a short response time is. To ensure the shortest possible response times, the file should be direct because a record can possibly be read with one disk access. Satisfactory response times also might be attained from an indexed file processed randomly by key because a record can be read with two disk accesses, one for the key and one for the data. If the direct file has numerous synonyms, multiple disk accesses can be required for each record, and this organization might not provide shorter response times than an indexed organization.

Transaction File Organization

Transaction files are less permanent than master files and typically are used to update master files. Transaction files are frequently logged in history files to keep records of business activities. An example of a transaction file is a cash receipts file for an accounts receivable application. An example of a more permanent transaction file is an open-item accounts receivable file. These transactions are usually maintained to show detail on reports such as statements sent to customers and aged trial balances.

Typically, transaction files entered from display stations in an interactive environment are direct files. The reason for using direct files is that the operator can usually expect a short response time when doing the following functions: paging through a file, adding records, deleting records, and reviewing all or part of the file.

The transaction file created by a WSU program, for example, is a direct file that has records separated *logically* by display station. Several display stations can enter transactions concurrently, and these transactions become mixed in the file. The transactions from a display station are chained by control information so that operators can access the records entered from his display station. This logical separation of records requires control records and control information in each transaction record. Subroutine SUBR22 allows an RPG II program to read records from a WSU transaction file.

Relative Record Number	Contents
1	Next available relative record number Last relative record number
2	W4 transaction 1
3	W1 transaction 1
4	W7 transaction 1
5	W1 transaction 2
6	W1 transaction 3
7	W4 transaction 2
8	W1 transaction 4
9	W7 transaction 2
10	W2 transaction 1
11	W1 transaction 5
	.
	.
	.

A direct transaction file might also be organized so that each display station has its own work area. For example:

Relative Record Number	Contents
1	Next available relative record number First relative record number W1 control record Last relative record number
2	Next available relative record number First relative record number W2 control record Last relative record number
	.
	.
	.
10	W1 transaction 1
11	W1 transaction 2
12	W1 transaction 3
	.
	.
	.
110	W2 transaction 1
111	W2 transaction 2
	.
	.
	.

Notice that a control record is required for the records entered from each display station. This direct file organization reduces the possibility of contention for the same sector of data, a condition described under *File Concepts* in Chapter 2; however, the number of records that can be entered from a display station is limited, and gaps can exist between the end of one section and the beginning of the next section.

Volatility of Files

The frequency of additions to and deletions from a file are important factors to consider when you choose a file organization. This disk activity is called volatility.

Highly volatile files might be direct files because a record can usually be added or deleted with fewer disk accesses than for other organizations, and fewer disk accesses should help shorten response times.

For example, adding records to indexed files requires (1) scanning the index, including added index entries, to ensure the record does not already exist; (2) reading the data area where the new record will reside; (3) writing the record; (4) writing the new index entry.

Adding records to a direct file might require (1) reading a control record to find the next available location, (2) writing the data, and (3) updating the control record. Updating the control record after each record addition makes programming for recovery easier but requires additional disk access.

The direct file might require disk space to allow for synonym records, and multiple disk accesses could be required for those records. Processing the direct file should be faster than processing a sequential file as an indexed file; however, in some cases an indexed file might be processed faster than a direct file that has many synonym records.

Refer to *Access Algorithms for Direct Files* later in this section for further information about synonym records.

Activity of the Files

Activity, which refers to the frequency that accesses are made to the file, is not as important a factor as how a file is used or how volatile the file is; however, activity should be considered when you choose a file organization.

Activity is usually referred to as a percent of the number of transactions to the number of records in the file. For example, if a file has 600 records and 1200 transactions are processed randomly per day, the activity is 200 percent.

As activity increases, consecutive processing becomes advantageous because of the chance that the record to process is available in a buffer and the record can be accessed without physical I/O activity. Therefore, very active files could be sequential and processed consecutively or could be indexed and processed sequentially by key. When an indexed file is processed sequentially by key, records added since the last key sort cannot be accessed, unless the IFILE attribute is specified.

A relatively inactive file might best be direct or indexed and processed randomly by key.

The total activity of a master file might be reduced by sorting a transaction file so that only one retrieval of a master record is needed for a group of transactions that have the same key.

RECORD DESIGN

After deciding which organization to use for a file, you can design its records and determine the file's size.

The applications determine what data is needed in the records. Study the applications then decide on the layout of the record. Layout means the arrangement of fields in a record. When you design a record, you should consider processing requirements of the record then determine each field's length, location, and name.

To illustrate these design considerations, a name and address file is described in the following text. Each record in the file contains the following data:

Field	Size (number of positions)
Customer number	6
Name	20
Street address	20
City and state	20
Record code	2
Delete code	1
Other fields	47
	116 Positions

Determining Field Size

Field size depends on the data in the field. The length of the data can vary, or all data in a field can be the same length. In the example, Name is 20 positions. The length of each customer's name varies, but 20 positions should be sufficient for most names without abbreviating them. Customer Number, however, is six positions, and all six positions are used in each record.

Numeric Fields

If the field is numeric, you should determine whether the field is to be in a packed or zoned decimal format. Packed format can reduce the amount of storage required.

Be sure to allow for the maximum length for dollar-amount fields, or a high-order position could be lost for exception conditions.

The maximum length of a packed field in RPG II is 15 digits (8 bytes). The following table shows the number of bytes needed for a specified number of characters in a packed field as compared to the number of bytes needed for that number of characters in a zoned decimal field.

Number of Characters	Number of Bytes Required	
	Zoned Decimal	Packed
1	1	1
2	2	2
3	3	3
4	4	4
5	5	5
6	6	6
7	7	7
8	8	8
9	9	9
10	10	10
11	11	11
12	12	12
13	13	13
14	14	14
15	15	15

The maximum number of numeric digits allowed for a numeric field is shown in the following diagram.

Programming Language	Numeric Digits Allowed	Notes
BASIC	14	Long precision
BASIC	6	Short precision
COBOL	18	
FORTRAN	17	Real*8 variable
FORTRAN	No limit	Decimal variable-no limits
RPG II	15	Packed or unpacked field

Key Length

The maximum alphameric key length is 29 positions. The maximum numeric key length is 15 bytes or 8 bytes if the key is packed. All relative record numbers in an addrout file are three positions long.

Alphameric Fields

No strict rules exist for determining alphameric field size. The major problem involves fields with variable length data. For example, if the name field is planned as 15 positions and a new customer name has 19 characters, a problem arises when this record is added to the file. To avoid this problem, try to estimate the maximum length of the data that will be contained in a field. Use this length to determine field size. The maximum length of fields for various languages is shown in the following table:

Language	Field Size
RPG	255 positions
BASIC	255 positions
COBOL	32767 bytes

Providing for a Delete Code

Your program can place a delete code in a record. Then, when the file is processed, your program can check for this code. For example, if a customer record becomes inactive, you may not want to process the record. Thus, a one-position field is included to provide for a delete code.

Records with a delete code are not physically removed from a file. To remove those records, you can use the ORGANIZE SSP procedure.

Providing Extra Space

Because record length is not yet established when the record is designed, you can allow for additions to the record. Although it may be difficult to plan what data might be added, you should reserve some extra space. For example, you might consider making the record length ten percent longer than initially required in order to allow for future additions.

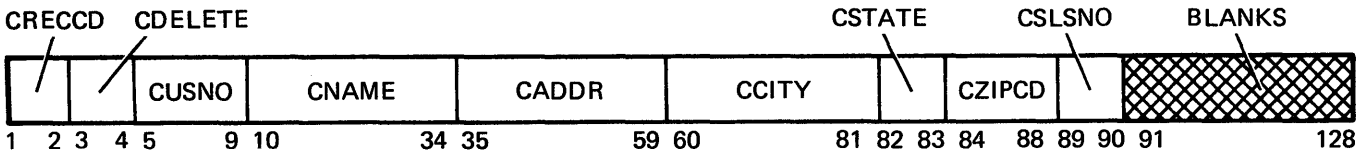
Naming Fields

An important consideration when choosing field names is that each name should be meaningful. Meaningful field names contribute to better documentation and help prevent misinterpretation or confusion during program writing. The language you use to write your program places restrictions on the length you can specify for your field names. The following table lists the maximum length your field names can be by programming language:

Programming Language	Maximum Field Length (characters)
RPG	6
COBOL	30
BASIC	8
FORTRAN	6

Documenting Record Layout

When record layouts are documented, your programs are easier to write. The following samples show the layout of a customer master record. Record layout includes the order of the fields in the record, the length of each field, and the name of each field. Notice that the field names follow the field-naming guidelines. The following diagrams show different ways of documenting your record layout. Included in this section is a sample form you can use for such documenting.



File name: CMAST
File organization: Indexed
Key: Customer number
Record length: 128

Field Description	Length	Decimal Position	Data Format	Location From	To	Field Name
Record code–MA	2		A	1	2	CRECCD
Delete code–D (blank if not active)	1		A	3	3	CDELETE
Customer number	6	0	N	4	9	CUSNO
Customer name	25		A	10	34	CNAME
Customer address	25		A	35	59	CADDR
City	22		A	60	81	CCITY
State	2		A	82	83	CSTATE
Zip code	5	0	N	84	88	CZIPCD
Salesman number	2	0	N	89	90	CSLSNO
Blanks	38		A	91	128	———

Record Length

Although field lengths within a record may vary, the field lengths for the same fields in each record in a file should be the same, and all records in a particular file must be the same length. Record length is the sum of the field lengths including reserved space. The maximum record length for a disk file is 4096 positions.

In the name and address file example in this section, the sum of the fields was set at 90 positions. However, record length was set at 128 to reserve 38 positions for data that might be needed at a later time.

INPUT/OUTPUT Record Description

File No. _____

Record Name Customer Record System _____ Page _____ of _____

File Name CMAST ☒ Disk Date _____

File Organization Indexed Sequence Customer Number Prepared by _____

Record Length 128 Key Customer Number Key Length 6

Created by _____ Used by ORDHDR Updated by _____

Values	Field Description	Field Name	Length	Decimal Pos.	Format	Location	
						From	To
MA	Record Code	CRECCD	2		A	1	2
D, blanks	Delete Code	CDELETE	1		A	3	3
000001-999999	Customer Number	CUSNO	6	0	N	4	9
	Customer Name	CNAME	25		A	10	34
	Customer Address	CADDR	25		A	35	59
	City	CCITY	22		A	60	81
	State	CSTATE	2		A	82	83
	Zip Code	CZIPCD	5	0	N	84	88
01-99	Salesman Number	CSLSNO	2	0	N	89	90
	Blanks		38		A	91	128

INPUT/OUTPUT Record Description

File No. ____________

Record Name ____________ System ____________ Page ______ of ______

File Name ____________ ☐ Disk ☐ Diskette Date ____________

File Organization ____________ Sequence ____________ Prepared by ____________

Record Length ____________ Key ____________ Key Length ____________

Created by ____________ Used by ____________ Updated by ____________

Values	Field Description	Field Name	Length	Decimal Pos.	Format	Location	
						From	To

RECORD BLOCKING

A block is the number of characters transferred as a unit of information between a disk file and the processing unit. Although only one record at a time is available for processing by your program, one or several records may be transferred in a block at one time.

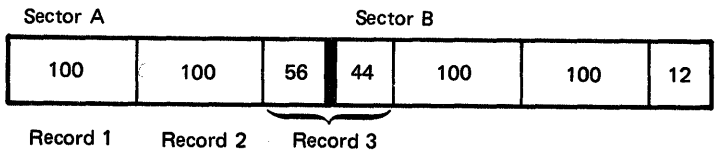
The block length is used to specify the amount of main storage used for an I/O buffer in the user program. Block length does not affect the way that records reside on disk, and the block length in a program does not have to match the block length specified when loading the file.

Block length is a multiple of record length. For example, if the record length is 64, the block length could be 256. Four records would be transferred at one time.

For efficient blocking, you should choose a record length that is either a multiple or submultiple of 256. For example, 512 would be a multiple of 256, and 64 would be a submultiple of 256 because it divides into 256 a whole number of times.

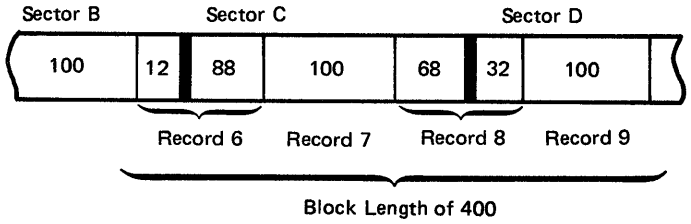
This choice is best because data is always transferred in sector increments, which are 256-byte increments, and you eliminate the chance of having records reside in more than one sector. For example, if the record length is 64 and the block length is 128, 2 blocks, which is one sector, would transfer with each physical I/O operation.

You can specify 100-character records as shown in the following example:



To process record 3, therefore, 2 sectors must be in main storage: sector A and sector B. The first 56 characters of record 3 reside in sector A; the remaining 44 reside in sector B. Thus, to process 100-character records with a block length of one hundred, 512 characters (2 sectors) must be available in main storage.

As another example, suppose you specified 100-character records with a block length of 400. Four 100-character records might span 3 sectors. To process your records in this case, 768 characters (3 sectors) might be required in main storage.



Blocking can be an advantage if you are likely to process multiple records in the block; by specifying a large block, you can reduce the physical I/O that occurs for a logical I/O operation.

For example, assume you read a file consecutively when the records are blocked 100 per block. The first get operation, a logical I/O operation, results in one relatively long read operation, a physical I/O operation, of 100 records. However, the next 99 get operations read from the buffer in main storage that holds the block of records and require no physical I/O.

For files processed randomly, you should not specify a large block length unless you are sure that more than one record will be processed in a block before another block is transferred.

For shared indexed or direct files that are processed randomly you should not block records because the entire block would be transferred for *each* input or output operation.

Finally, you might choose not to block records if you are trying to keep a program from getting too large.

PHYSICAL I/O AND LOGICAL I/O

Physical I/O operations are those operations that cause disk read and write operations to occur. These operations take time because disk arm positioning and moving is usually required. Therefore, during system design you normally plan to minimize physical I/O operations in order to improve response times and system performance.

Logical I/O operations are get and put operations that access records. The number of physical I/O operations that result from logical I/O operations is affected by the following factors:

- Record blocking
- Access method
- Storage index
- Sequential processing
- File sharing
- Buffer sharing

Blocking Records to Minimize Physical I/O

By specifying a relatively large I/O buffer for a program, you can often minimize the physical I/O activity for a set of logical I/O operations. A large I/O buffer reduces physical I/O activity for files that are not shared and for shared files that are processed consecutively or sequentially by key for input only.

For example, assume you specify a block of 100 records and use the consecutive input access method. The first get operation causes a disk read operation to occur that reads 100 records into the buffer. The next 99 get operations read from the buffer and require no physical I/O operations.

Refer to *Record Blocking* in this section for further explanation.

Access Method

Another factor that affects the amount of physical I/O activity is the access method that you use. When a file is accessed with an indexed access method, each logical I/O operation results in a logical I/O operation that processes the index entry and a logical I/O operation that processes the data record. The amount of physical I/O activity that occurs depends on whether the contents of the index buffer can be used for index entry processing and whether the contents of the data buffer can be used for data record processing.

For example, assume that the logical I/O operation is an indexed sequential get and the index buffer already contains the next index entry. In this situation, physical I/O is not required to process the index entry, which can be read from the buffer by the logical operation. After the index entry is processed, the data buffer is searched to find the associated data record. If the record is in the buffer, the record can be read by a logical operation, and physical I/O does not occur.

In the previous example, either one or two disk read operations would have been required if either the index or the record had not been in the buffers.

Storage Index

For COBOL, RPG II, WSU, BASIC, and Basic Assembler programs you can request that the system create a storage index, which is an in-storage index to the actual file index on disk. (The storage index is called a master track index in some System/34 publications, and a key work area in BASIC.) A storage index can significantly reduce the amount of physical I/O activity required to process an indexed file because the storage index enables the system to go more directly to a record you want.

When a storage index is not used, the system sequentially searches the file index until it finds the entry for a requested record. This search often requires several I/O operations. When you request a storage index, the system divides the file index into segments and then creates a storage index, which points to each of the segments. The storage index eliminates needless searching by directing the system to the index segment containing the entry for the requested record.

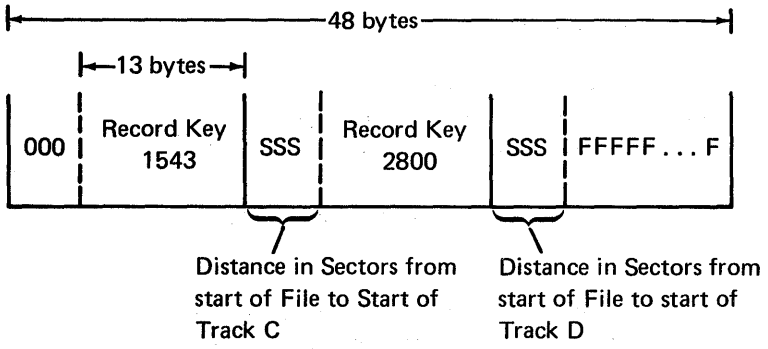
If the space allocated for the storage index is large enough, the system divides the file index into segments one track long. If enough storage index space is not allocated for this optimal segmentation, the system divides the file index into larger segments. Each storage index entry contains (1) the lowest key field from the next segment in the file index and (2) a sector address that points to a segment of the file index.

The sector addresses in the storage index are relative sector addresses which represent a displacement sector address from the start of the file.

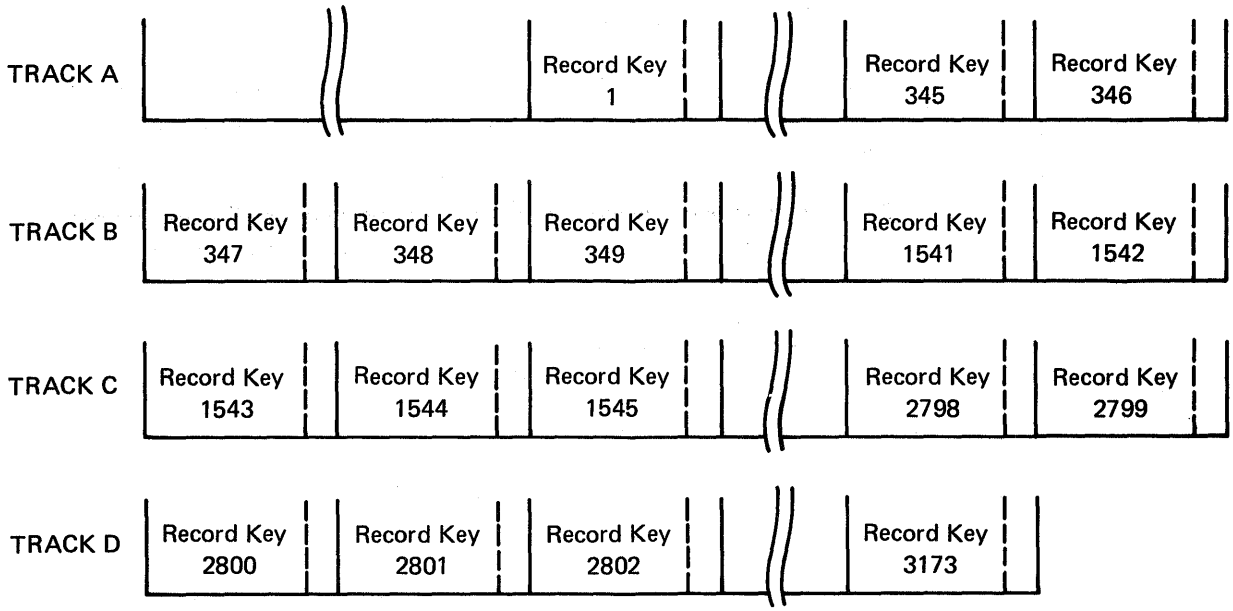
Note: The storage index is supported by the SETLL instruction for RPG II and the START instruction for COBOL.

The following chart shows the storage index for a file index that occupies four tracks on disk. If a program requests the record with key 1541, the system searches the storage index and determines that the requested record is on track B. The system then scans the index entries on track B until the entry for record 1541 is found. The system can then chain directly to the data record.

Storage Index for File INDEXT



File Index for File INDEXT



To calculate the amount of space required for the optimal storage index, you should first calculate the number of tracks used by the file index. The number of tracks used by the file index is related to the capacity of your disk. You will use different numbers to calculate the file index if you have more than 27.1 megabyte disk capacity. To determine the number of tracks used by the file index and the amount of space required for the storage index:

1. Use the STATUS command to determine disk capacity.

Is the disk capacity greater than 27.1 megabytes?

   a. If yes, the file index will have 64 sectors per track.
   b. If no (that is, if the disk capacity is less than or equal to 27.1 megabytes), the file index will have 60 sectors per track.

2. Use the CATALOG procedure to determine the number of records in the file.

3. Add three (3) to the key length to determine the length of an entry in the file index. (Each entry in a file index is composed of the record key plus the 3-byte address of the record in the file.)

4. Divide 256 by the entry length to determine the number of keys in each sector. Drop the remainder.

5. Divide the number of records in the file by the number of keys in each sector to determine the number of sectors in the index. Round off the result.

6. Divide the number of sectors by either 60 or 64 to determine the number of tracks in the index. Round off to the nearest whole number any fractional result.

7. Multiply the number of tracks in the index by the length of an entry in the file index to determine the optimal index size.

The following two examples, using an indexed file containing 8000 records with 9-byte keys, show the calculation of the index size. The examples show the effects of different disk capacities on the optimal size of the storage index.

	Disk capacity greater than 27.1 megabytes	**Disk capacity less than or equal to 27.1 megabytes**
1		
2	8000 records	8000 records
3	9 + 3 = 12 bytes for the file index entry	9 + 3 = 12 bytes for the file index entry
4	256/12 = 21 keys in each sector	256/12 = 21 keys in each sector
5	8000/21 = 381 sectors in the index	8000/21 = 381 sectors in the index
6	381/64 = 6 tracks in the index	381/60 = 7 tracks in the index
7	12 * 6 = 72 bytes for the storage index	12 * 7 = 84 bytes for the storage index

If the application program cannot accommodate the amount of storage required for the optimal sized storage index, you can specify a smaller size, but this smaller size results in a correspondingly smaller improvement in performance.

Sequential Processing

If record processing is sequential relative to the physical order of records on disk, a performance advantage can be gained. For example, assume 50 records per block and sequential processing is used. Physical I/O for data might be required only once for each set of 50 logical requests.

File Sharing

File sharing affects the amount of physical I/O activity for each logical I/O operation. For most access methods, each logical I/O operation causes at least one physical I/O operation.

Exceptions to this guideline occur for consecutive input-only processing and indexed sequential input-only processing, which do not always require physical I/O activity for each logical I/O operation.

Shared I/O

Shared I/O allows two or more files in one job to share the same buffers and causes a large amount of physical I/O activity because buffers must be reread for each logical I/O operation.

ACCESS ALGORITHMS FOR DIRECT FILES

A key to designing and implementing a direct file is defining an access algorithm that satisfies the processing requirements for the file while preserving the advantages of direct files.

Determining an Access Algorithm

An *access algorithm* is whatever fixed (programmed) method is used to determine the position to be occupied by each record. The algorithm can be simple or complex. In any case, the algorithm must yield a positive, whole number as a relative record number.

In the simplest case, relative record numbers are assigned sequentially. The first record placed in the file has relative record number 1, the second record has relative record number 2, and so on.

In another simple case, a control field in each record is used as its relative record number. For example, loan number 3456 could be used without change as relative record number 3456. Another example of a direct technique is using direct files to store large arrays of data. If element 10 is desired, then the tenth record in the file is read. A control field should be used directly as a relative record only if there is not an excessive number of unused values within the range of values for the control field. If there are too many unused values and, therefore, unused record positions, an algorithm should be defined to reduce the size of the file.

A formula can be used as an algorithm to determine the record number. For example, if loan numbers start with 1001, then loan number 3456 could be relative record number 2456 (3456 minus 1000). The formula can be as complex as you need to make it. Refer to *Examples* later in this chapter for more information and examples.

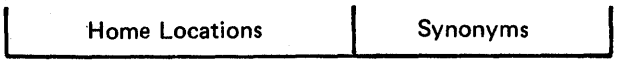
A control field that contains alphameric data could also be used. An algorithm must convert the alphameric data to a relative record number. Refer to *Handling Synonym Records* later in this chapter for an example of using a customer name as the control field.

The choice of an access algorithm and, ultimately, the decision whether or not to use a direct file is usually based on how well synonym records can be handled. A *synonym record* is a record in a direct file whose control field yields the same relative record number as another control field. If the handling of synonyms requires a significant number of additional disk accesses, one of the important advantages of the direct file is lost. Also, because the access algorithm and the synonym code must reside in each program that uses a direct file, a risk is involved: if the algorithm and synonym handling are revised, you might need to rebuild files and modify all the programs that use those files.

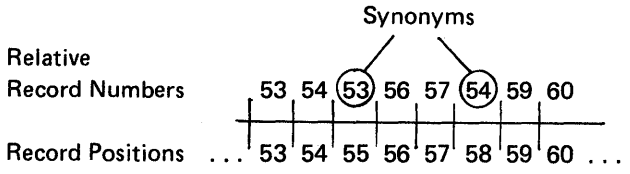
Handling Synonym Records

Synonyms can be handled in many ways. Some of the common ways are:

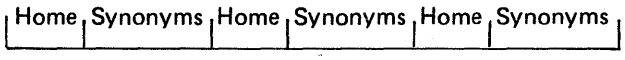
- Place synonyms in a separate part of the file, following the home locations, which are the locations used for home records. A *home record* is a record that is stored in the location indicated by its relative record number.



- Place synonyms in the next available blank location, closest to the home location.

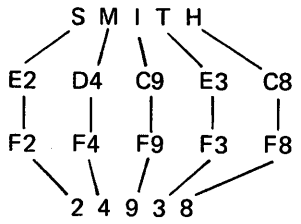


- Place synonyms in an area, next to the home location, that is reserved for synonyms.



In the first two methods, the record in the home location must contain a pointer to the synonym record location. If two or more synonyms exist for a home location, the first synonym contains a pointer to the second synonym, and so on.

In the third method, synonyms are close to the home location. For example, assume the control field for a file is the first five characters of the customer's name. The file contains space for 40000 records and allowance for three synonyms for each home record. The customer's name is converted to a decimal value as follows:



(EBCDIC code)

(zoned decimal)

(decimal)

The decimal value is then divided by 9999:

24938 ÷ 9999 = 2.4940

Ignoring the whole number of the quotient, you would calculate the home location as follows:

(4940 x 4) + 1 = 19761

Because many Smiths may be in the file, the program may have to read records 19761, 19762, 19763, and 19764 to find the correct Smith. If extra synonyms are required, the third synonym could point to the next available space in the file (possibly the next home location will not require all its synonym locations). Another possibility, to reduce the number of synonyms, would be to accept six or more characters from the customer name.

Examples

The following examples illustrate direct file approaches.

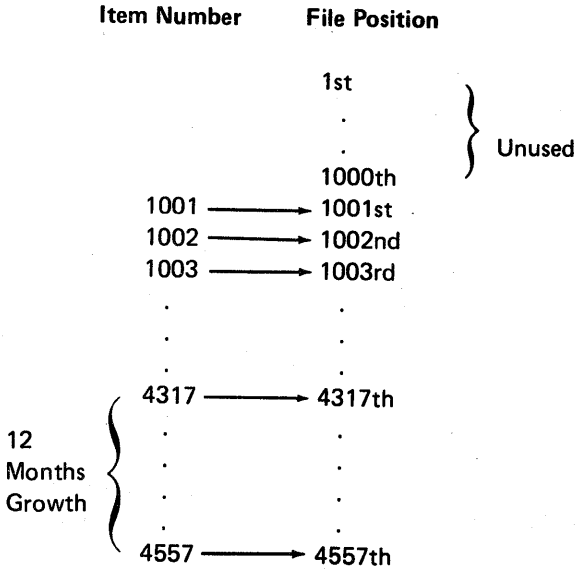
Example 1

In this example, the major goals are to build a file in which (1) the records can be accessed with an average of slightly more than one disk access, (2) the amount of disk space used for the file does not contain excessive unused space, and (3) the file can grow and easily accommodate new records.

Defining the Algorithm

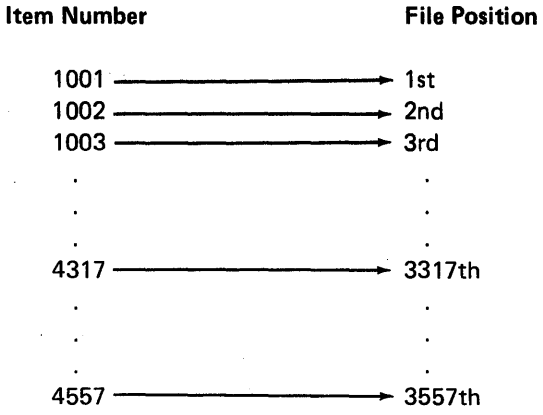
In this example, an indexed item file is to be converted to a direct file for an online order entry application. The key field is a five-digit item number; four digits are assigned by the user, and the fifth digit is a check digit. The four digits start with 1001, and the user merely assigns the next sequential number to new items. Deleted item numbers are not reused until item number 9999 has been taken. Approximately 20 new items are added per month, and four items are dropped. The highest current number is 4317, but the file contains only 2812 items.

As a first approach, the algorithm could be stated this way: the direct file position for each record shall be equal to the four-digit item number. Assume that the new record will be a few bytes larger than the old record and that the file will also accommodate 12 months of growth before reorganization. The algorithm would require a file containing 4557 record positions. The mapping of items to direct file positions would appear as follows:



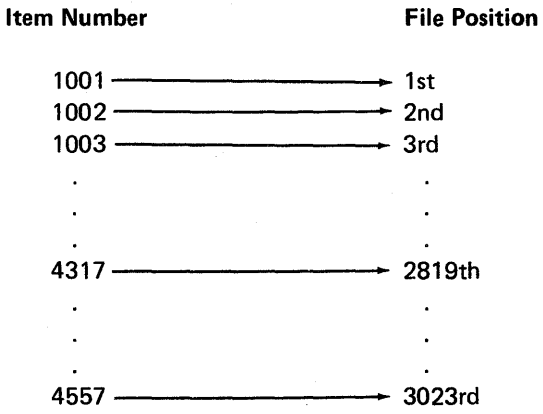
This first approach, while yielding no synonyms, uses only two-thirds of the record positions, and most of the unused space is at the beginning of the file.

Assume the algorithm is revised to state: the direct file position for each record shall be equal to the four-digit item number minus 1000. The file requires 3557 positions with the following mapping:



This approach, also yielding no synonyms, uses 85 percent of the record positions; the unused portion is embedded randomly within the file where items have been dropped. Although each record only requires one disk access, the file size still is 15 percent larger than the data portion of the file when organized as an indexed file. The algorithm can be further revised.

Now assume the algorithm states: the direct file position for each record shall be found by subtracting 1000 from the four-digit item number, multiplying the difference by 0.85, and half-adjusting the result. The file will occupy 3023 positions with the following mapping:



This approach uses 99 percent of the record positions, and the file size is only 1 percent larger than the indexed data. It has, however, introduced the possibility of synonym records; item number 1004, if it exists, will also be assigned to direct file record position number 3 (same as 1003). Similarly, item numbers 4316 and 4317 conflict, as do 4556 and 4557. Thus, the refinement of the algorithm to meet the second major goal, minimum file space, may now have affected the first goal, minimum disk accesses, because synonym records will take a minimum of two accesses.

Various methods of handling synonyms exist. Whatever method is used, it must accomplish two goals: minimum accesses and minimum file space. The more immediate goal is to define (program) the manner in which a record will find an alternate position when its first location choice is filled.

Further analysis of the previous item file example might offer some suggestions for synonym handling. Note that a synonym in that example occurs about once in seven records.

The previous algorithm causes the following mapping (asterisks identify synonyms):

Item	Position	Item	Position	Item	Position
1001	1	1009	8	1017	14*
1002	2	1010	9*	1018	15
1003	3*	1011	9*	1019	16
1004	3*	1012	10	1020	17
1005	4	1013	11	1021	18
1006	5	1014	12	1022	19
1007	6	1015	13	1023	20*
1008	7	1016	14*	1024	20*

Approximately one in seven item numbers is unused because of deleted items; the file is only 86 percent full. Thus, you might expect to find an unused position in the direct file with about the same frequency as the synonyms occur.

Assume the method of handling synonyms can be stated: a synonym record will be placed in the next higher numbered position that is unused. Because the file uses only 85 percent of the range of numbers, 15 percent of the numbers will not be used because they are deleted. However, the deleted numbers are randomly distributed through the entire range of numbers. Thus, some positions will be available in the file for synonym records. About every seventh number will be a synonym. Assume that of the first 40 item numbers, items 1007, 1008, 1015, 1017, 1020, and 1039 are among those deleted numbers.

Item	Position	Item	Position	Item	Position	Item	Position
1001	1	1013	11	1026	22	1036	31
1002	2	1014	12	1027	23	1037	**
1003	3	1016	14	1028	24	1038	32
1004	6	1018	15	1029	25	1040	34
1005	4	1019	16	1030	26		
1006	5	1021	18	1031	**		
1009	8	1022	19	1032	27		
1010	9	1023	20	1033	28		
1011	13	1024	33	1034	29		
1012	10	1025	21	1035	30		

Note the following:

- Item number 1031 will occupy some position numbered greater than 34.
- Item number 1037 will occupy a higher numbered position than will item number 1031.
- Record positions 7 and 17 are unused.
- After accessing a record, the program will have to verify that the record is the one that the program really wants; if it is not, the program must access a synonym.
- There will not be more than two items with the same relative record number; thus, most records require no more than two disk accesses.

  Note: This example assumes that records are loaded into home locations before synonym records are loaded in a second run; this example also assumes that there will be few added records. If records are added after the home records and synonyms are loaded, the home locations for the added records may be occupied by a synonym. Thus, the added record becomes a *pseudo synonym*. If many records are added, most will have to be handled as synonyms. In this situation, the technique described here may be less useful because performance tends to be degraded as records are added.

In this synonym-handling technique, the average synonym should be close to the first position searched. Thus, a second access is necessary approximately 15 percent of the time, and this access should find the record not too distant from the home location.

At this point, the file should be loaded (home positions only), and the synonyms added in a second pass. As the synonyms are added in the next available higher numbered position, a synonym pointer in the home record will have to be updated to point to the synonym record position.

Example 2

Assume a customer master file contains three types of records (A, B, and C) for three types of customers. These records are in an indexed file in which type A records have keys–customer numbers from 10000 to 49999; type B records are numbered from 60000 to 79999; and type C records from 90000 to 99999. Each type of record is arranged alphabetically by customer name.

The file was first loaded with approximately 500 alphabetized type C records, followed by 1000 alphabetized type B records, and finally about 3000 alphabetized type A records.

Additions have been made at the end of the file in the following manner: first, the added record type is determined–A, B, or C; then it is assigned an unused customer number that corresponds to the alphabetic sequence of the customer name according to a listing of the file. When first loaded, the contents of the file were as follows:

Record #0001	Customer #90000	Type C (alphabetical by customer name)
Record #0002	Customer #90020	
Record #0003	Customer #90040	
.	.	
.	.	
Record #0467	Customer #60020	Type B (alphabetical by customer name)
Record #0468	Customer #60040	
Record #0469	Customer #60060	
.	.	
.	.	
Record #1592	Customer #10000	Type A (alphabetical by customer name)
Record #1593	Customer #10013	
Record #1594	Customer #10026	
Record #1595	Customer #10039	

The file originally contained 4725 records; space was allowed for 6000. Eighteen months later, the file contains 5638 records.

An analysis of the file indicates the following:

- The file is experiencing about 12 percent annual growth and should probably be planned for about 6600 records to meet one year's requirements.
- Customer numbers 10000-50000 are 8 percent used, and the other numbers are 5 percent used.
- Synonym records should be kept as close as possible to the home location.
- The best file design solution might be more than one file and more than one type of file organization.
- If all the customer numbers will be in one file, an algorithm must take into account the necessity of loading type C customers at the front of the file, followed by types B and A.
- The ratio of A:B:C types is about 6:2:1.

A trial algorithm might try to accomplish the following mapping:

Customer Number	Type	File Record Number
90000-99999	C	0001-0733 (1/9 x 6600 = 733)
60000-79999	B	0734-2200 (2/9 x 6600 = 1467)
10000-49999	A	2201-6600 (6/9 x 6600 = 4400)

In order to accomplish the mapping, the algorithm must:

- Convert customer numbers 90000 to 99999 into a set of relative record numbers from 1 to 733
- Convert customer numbers 60000 to 79999 into a set of relative record numbers from 734 to 2200
- Convert customer numbers 10000 to 49999 into a set of relative record numbers from 2201 to 6600

One method of doing these conversions is as follows:

- If the customer number is greater than 89999, subtract 89999 from it; then multiply the difference by 0.0733 (the ratio of 733 positions to 10000 numbers), and use the half-adjusted product as the record position.

- If the customer number is less than 50000, subtract 9999 from it; then multiply the difference by 0.11 (the ratio of 4400 record positions to 40000 record numbers), add the half-adjusted product to 2200, and use the sum as the record position.

- For all other customer names (60000 to 79999), subtract 59999 from the number, multiply it by 0.0733 (the ratio of 1467 record positions to 20000 numbers), add the half-adjusted product to 733, and use the sum as the record position.

The synonym-handling technique might be the same as in *Example 1*.

The test of success might be to implement the algorithm/synonym-handling technique by loading the file. Then the success can be measured by another program that attempts to retrieve all records and counts the number of accesses necessary. The results of the second program dictate whether modifications are necessary or desirable. To further test the file, a sample program can be run in an interactive environment to see whether response times at the display stations are acceptable.

Example 3

Other master files might have altogether different uses and for that reason use different techniques. Consider a rate file in a telephone revenue accounting application wherein one record exists for every *from-to* location in the United States. A call made from number (507) 286-5688 to (518) 392-5536 would require the retrieval of a rate record from the master file that would have a key of 507286518392. How can such a number be equated to a relative record position on a direct file?

One algorithm might be to multiply the numbers 507286 and 518392 and use the second, fourth, sixth, eighth, and tenth digits of the product as the relative record position. This technique might yield a random distribution across a file for approximately 100 000 records. Another algorithm might be to take the second, fourth, sixth, eighth, and tenth digits from the 12-digit key. Thus, the first algorithm might locate the rate record in relative position 69301 (262973004112); the second algorithm might place the same record in position 02613. Some records, for a given billing location, would be far more active than the majority of the records. These very active records might be placed in a separate file, which may or may not be direct.

The techniques described in the previous paragraph are randomizing techniques. Many randomizing techniques are employed by users of direct files. Regardless of which technique is used, the concept and approach should be well documented in each program that uses the technique.

Application Design

After the applications for the System/34 have been chosen, you can plan which programs should be designed and implemented first. You might consider the following factors to help you decide upon the initial programs:

- Which application best justifies the cost of the System/34? An accounts receivable application or an order entry application might benefit the business most quickly.
- Which application makes the user departments more productive and the users' jobs easier? Inquiries from display stations are typically programs that can help the users do their jobs more effectively.
- Which application can be designed and implemented in the least amount of time? Again, applications that have numerous inquiries are usually easiest to plan.
- Which application are you most familiar with? If possible, starting with an application that you have implemented or operated on another system is usually a good idea.

A good design technique, therefore, is to begin by designing the easiest programs and gradually add the more complex programs. Inquiry programs are usually a good starting point because they are relatively simple and operators can use them productively with minimal instruction. More complex programs can be added as confidence in programming and operating the System/34 grows.

Most applications for System/34 have both batch and interactive programs. Interactive programs communicate with one or more display stations; batch programs do not have this interaction.

Interactive programs can be classified as data entry, inquiry, or file update. The following examples show typical interactive programs:

Function	Program	Application
Data entry	Order entry	Order entry
Inquiry	Open order inquiry	
File update	Inventory allocation	
Data entry	Cash receipts entry	Accounts receivable
Inquiry	Accounts status inquiry	
File update	Open accounts receivable items	
Data entry	Inventory receipts/adjustments entry	Inventory control
Inquiry	Inventory status inquiry	
File update	Vendor code changes	

DATA ENTRY PROGRAMS

Data entry programs allow one or more operators to enter data directly into the system via display stations. Data entry programs involve nearly continuous operation of display stations. Operators enter transactions such as order information, cash receipt information, or inventory receipt/adjustment information, and programs process them or save them for later processing.

Typically you will need to choose data entry programming methods. The Data File Utility (DFU), Work Station Utility (WSU), and RPG II Program Product are three methods that offer similar data entry functions. The following table and descriptions compare these data entry methods. These three methods are not the only ways to code interactive data entry programs. For example, COBOL, BASIC, FORTRAN, and Basic Assembler programs can interact with display stations.

DFU Data Entry Programs

DFU programs are generally best for high-volume data entry with minimal operator guidance and no editing other than modulus 10 and modulus 11 self checking. DFU can create an indexed file or a direct file from the entered data. Programming is easily done by answering a series of prompts.

DFU programs are SRT programs. Multiple operators, if they run the same program, use separate copies of the program and unique or shared transaction files. Refer to the *System/34 DFU Reference Manual* for a complete description of the functions of DFU.

To allow several operators to enter the same type of data using common DFU specifications, you can create a procedure that contains the required DFU command. The file name in this command should be the file name concatenated with the ID of the display station that calls the procedure. This technique results in unique files for each operator. For example, creating an order file from two display stations could be allowed by using ORD?WS? for the file name. Two files would be created: ORDW1 from display station W1 and ORDW2 from display station W2.

WSU Data Entry Programs

WSU programs optionally create a direct transaction file from a master file or files and the data entered from one or more display stations. WSU programs can be front-end entry programs for programs that do final editing, processing, updating, and printing. WSU is designed for efficient display station data entry and processing and, therefore, does not provide printed output.

WSU programs are always MRT programs, which means they can be used concurrently by multiple operators. The MRT program can use input from operators, from master files, and from results of processing within the program to create and maintain one transaction file. The program can also add records to and update any master files that are used by the program.

Two or more different WSU programs can be running at the same time and can share master files. For example, payroll input, job costing input, and accounts payable can be handled by three separate WSU programs that run concurrently. Sharing of transaction files, however, is not supported.

The transaction file that a WSU program creates is a direct file, and the records are partitioned logically by display station. WSU formats the transaction file with special header and trailer records to identify the records. WSU protects each partition so that records entered from one display station cannot be read or modified from another display station unless a special WSU function is enabled. This special function allows you to read and modify records entered from another display station. For more information about this special WSU function, refer to Chapter 14 of the *Work Station Utility Reference Manual*.

Because of the headers and trailers, the transaction file cannot be input to a follow-on program until this control information has been removed or handled in the program. RPG II provides a subroutine, SUBR22, that can be used to prepare the transaction file for processing. Refer to the *System/34 RPG II Reference Manual* for a description of SUBR22.

WSU also provides support for removing the control information. A rebuild procedure for WSU allows you to copy a WSU transaction file and change it to an indexed or sequential file. The new file can then be processed by a subsequent program.

An extract procedure can copy specific records to another file or remove blank records from a WSU transaction file. The extract can also be used to create a WSU transaction file by adding control information. This function allows you to create a file on a separate data entry device and then update the file using WSU.

A WSU program, using an indexed random or a direct access processing method, can read from and write to as many as 20 master files. These files can be indexed, direct, or sequential files. Master files can be reviewed, updated, and added to by operators if the programmer codes these functions in the program.

WSU programs can do more edit checking than DFU programs. Some checking is done by the display station; for example, numeric data entered in an alphabetic field will cause an error to be flagged, and the operator can reenter the field. Some checking can be specified by the programmer on display screen specifications; for example, mandatory fill, mandatory enter, and self-check fields can be defined on these specifications. Additional checking can be done within the program; for example, the calculations for a display can check for valid entries within a range of values and reshow the display with the cursor positioned at the field in error.

Refer to the *System/34 WSU Reference Manual* for a complete description of WSU programs.

RPG II Data Entry Programs

RPG II provides three methods of coding data entry programs:

- WORKSTN file
- CONSOLE file
- SET/KEY logic of the KEYBORD file

Programs that use a WORKSTN file have the most display design flexibility because these programs use display screen formats defined by the programmer. For example, field attributes and a variable starting line number can be specified. Also, multiple items per display can be programmed.

Programs that use a CONSOLE file are easy to code. The format of displays are determined by RPG II when the program is compiled. Multiple items per display are allowed and require no additional programming.

Programs that use SET/KEY logic provide field-by-field data entry. Data is entered on the bottom line of the display and, when entered, rolls up one line to allow the next field to be entered on the bottom line.

Refer to the *System/34 RPG II Reference Manual* for descriptions of coding data entry programs via the previous three methods.

Function/Feature	DFU	WSU	RPG				
			KEYBOARD	CONSOLE	WORKSTN	COBOL	BASIC
Display layout							
Field attribute selection	No	Yes	No	No	Yes	Yes	Yes
Variable starting line number	No	Yes	No	No	Yes	Yes	Yes
Multiple items per display	No	Yes	Not recommended	Yes	Yes	Yes	Yes
Display associated with record type	Yes	Programmable	No	Yes	Programmable	Programmable	Programmable

Function/Feature	DFU	WSU	RPG	COBOL	BASIC	FORTRAN
Printed output	Yes	No	Yes	Yes	Yes	Yes
Master file access	LIST only	Yes	Yes	Yes	Yes	Yes
Multiple user (MRT)	No	Yes	Yes	Yes	Yes	No
MRT code handled	–	Most	Some	Programmable	Programmable	–
Transaction file	Indexed, sequential, direct	Partitioned direct	User-coded	User-coded	–	–
Maximum record length	512	4083	4096	4096	4096	4096
Review mode	Yes	Yes	Programmable	Programmable	Programmable	Programmable
Insert mode	Yes	Yes	Programmable	Programmable	Programmable	Programmable
Control over operator modification	No	Yes	Yes	Yes	Yes	Yes
Self-check numbers	Hardware feature	Hardware feature	Hardware feature	Hardware feature	Hardware feature	Hardware feature
Maximum alphameric field length	60	256	256	4096	255	4096
Calculations	Limited	Yes	Yes	Yes	Yes	Yes
Number of result fields	24[1]	Many	Many	Many	Many	Many
Arrays	No	No	Yes	Yes (3-D)	Yes (2-D)	Yes (3-D)
Edit checking	No	Yes	Yes	Programmable	Programmable	Programmable
Batch totals with insert and review mode adjustments	Yes	Programmable	Programmable	Yes	Yes	Yes
Time of day	Yes	Yes	Yes	Yes	Yes	Yes
Date	Yes	Yes	Yes	Yes	Yes	Yes
Local data area access	No	Yes	Yes	Yes	Yes	Yes
User switch access	No	Yes	Yes	Yes	Yes	Yes
Roll keys handled	Yes	Yes	Programmable	Programmable	Programmable	No
Dup key handled	Yes	Not available	Programmable	Programmable	Programmable	No
Roll portion of screen	No	No	No	Yes	No	Yes
Write on message line	No	Yes	No	Yes	Yes	No
Allow help key	No	No	Yes	No	Yes	No
Read message member via format output	No	Yes	Yes	Yes	Yes	Yes
Explicit open/close	No	No	No	Yes	Yes	Yes
User indicators	No	Limited	Limited	Yes	Yes	No

Note: Yes means the function or feature is available for the programming method; No means the function or feature is not available.

[1]Less if more than one factor is used per result field.

The Badge Reader as a Data Entry Device

You can use your badge reader as a means of entering data into the system. The data is encoded onto a magnetic stripe which is part of the badge that is read by your badge reader.

The encoding of this data onto the magnetic stripe involves special data encoding considerations. Contact your local IBM branch office for more information about encoding data for use with magnetic stripe badges.

Editing in Data Entry Programs

Data editing at the time data is entered into the system should be a primary function of an online application. The same *basic* edit functions apply to data regardless of the application involved. This section addresses some major editing problems that you could encounter.

One or more of the following edit checks may be specified for any given data field.

- Field editing
  - Mandatory field entry
  - Every field required
  - At least one field required
  - Only one field required
  - Default value editing
  - List check editing
  - Range check editing
  - Mandatory fill
  - Self checking
  - Adjust/fill
  - Duplication
- Date editing
- Compatibility editing
- Table look-up editing

The following sections provide a detailed discussion of some of the types of edit functions that could be considered.

Field Editing

Field editing performs basic data characteristic validation for data fields within a transaction. Some of the key types of data characteristics possible are:

- All characters must be blank or alphabetic.
- All characters must be blank or numeric.
- All characters must be blank, alphabetic, or numeric.
- All characters must be numeric.
- All characters must be numeric, with leading blanks optional. Field must not contain embedded blanks.
- All characters must be alphabetic or numeric.
- All characters must be alphabetic. Field must not contain leading or embedded blanks.
- All characters must be alphabetic, numeric, blanks, or special characters.
- All characters must be alphabetic, special characters, or blanks.
- All characters must be numeric, special characters, or blanks.
- All characters must be special characters or blanks.
- All characters must be special characters.
- All characters must be alphabetic, numeric, or blanks, with no leading blanks.
- All characters must be alphabetic, numeric, or blanks, with no trailing blanks.
- All characters must be alphabetic, with no embedded blanks. Leading and trailing blanks are permitted.
- All characters must be numeric, with no embedded blanks. Leading and trailing blanks are permitted.
- All characters must be numeric, alphabetic, or special characters, with no embedded blanks. Leading and trailing blanks are permitted.
- All characters must be alphabetic or special characters, with no embedded blanks. Leading and trailing blanks are permitted.
- All characters must be numeric or special characters, with no embedded blanks. Leading and trailing blanks are permitted.

- All characters must be special characters, with no embedded blanks. Leading and trailing blanks are permitted.

- All characters must be alphabetic, numeric, with no embedded blanks. Leading and trailing blanks are permitted.

Mandatory Field Entry: Specifies that a value must be entered for a field. Blank fields that have been designated as required fields are invalid. Other types of editing may also be done on required fields. If a field is to have a default value, mandatory field entry should not be specified.

Required Entry of Every Field: Specifies that all fields in a logical group must be entered if any one field of the group is entered. This capability might be used, for example, on a mailing address to specify that if any part of the address–street, city, or zip code–is entered, then all the mailing address fields are required.

Required Entry of at Least One Field: Specifies that at least one field in a logical group of fields must be entered. This editing might be used, for example, on a list of the reasons a student has withdrawn from a university, to ensure that at least one reason for the withdrawal has been entered.

Required Entry of Only One Field: Specifies that only one field in a logical group may be entered. If a group of fields is mutually exclusive, but one field is required, this type of editing would be used. This capability might be used, for example, to prevent the scheduling of more than one surgical procedure at a time per operating room.

Default Value Editing: Moves specified default values into a data field if nothing is entered in that field. If data is entered in the field, the default value is ignored, and normal editing is performed.

List Check Editing: Specifies a list of valid values for a data field. If the content of a field is not equal to one of the values in the list, the field is invalid.

An optional feature is to have a default value substituted for the data entered if the value of the data entered is not in the list.

Range Check Editing: Determines whether the value contained in a given data field is between predetermined high and low boundaries established for that particular data field.

The range edit function could specify one of three possible conditions for data fields that have been range edited: (1) the data value is lower than the specified range, (2) the data value is higher than the specified range, (3) the data value is within the specified range.

Mandatory Fill: Ensures that all positions in the field are entered. Thus, for example, entry of only four numeric digits in a five-position code field would be invalid. A field designated as variable, such as a 25-position name field, need not require that all the positions be filled to be considered valid.

Self Checking: Specifies that the data entered in the field is checked by a modulus 10 or modulus 11 algorithm after the field is entered.

Adjust/Fill: Specifies that data entered in a field can be right-adjusted and that unused positions are filled with zeros or blanks.

Duplication: Indicates that the Dup function key can be used to fill the position of the cursor and the positions in the field to its right with the duplication character, which is * (hexadecimal 1C). The user program must check for these characters and place the appropriate duplicated data in the field.

Date Editing

Date editing should validate date fields within a transaction.

Either of two types of date editing may be specified for date fields. The first type of date edit determines, in general, whether or not a given date field is valid: the month is between 01 and 12; the day is between 01 and 28, 01 and 29, 01 and 30, or 01 and 31 depending on the month and whether or not it is a leap year; and the year is numeric and within a predetermined range.

The second type of date editing validates the date field as above and, if the date is valid, determines whether the date is on or after the current system date. This form of date editing is used when the date wanted must be either the current date or some date in the future.

Compatibility Editing

Compatibility editing should ensure that designated data fields are compatible with each other by cross-checking their respective contents. In other words, the data within a given field is valid only if another field contains a specific value or is within a specified range of values. If this is not the case, then an error condition exists, and the first field is flagged as having failed compatibility checking. Therefore, even if a field passed field editing, it may fail overall editing due to incompatibility.

Table Lookup Editing

Table lookup editing should provide the ability to alter the contents of a field based on a conversion table established by the user. This is basically a one-for-one replacement of data according to the conversion specifications contained in the table. The table contains the values of the field to be converted and corresponding substitution values used to replace the original values.

When a match is found in the conversion table for a given field value, the replacement value is placed in the edit result table, and table lookup is considered to be successful. If the field contains a value not in the conversion table, then the field is considered invalid and is flagged.

An optional feature of table lookup editing is to have a default value substituted for the data entered if that entry is not in the conversion table.

Summary of Editing

Collectively, the above edit capabilities provide a powerful tool to the user of an online system and although any or all the edits may be performed on any given data field, the extent of editing is at the discretion of the user. Field data editing is usually performed initially because this validates a data field before additional editing is performed. Additional types of editing are usually not performed if field data editing indicates a given data field is invalid.

INQUIRY PROGRAMS

Inquiry programs are the simplest of the types of interactive programs to design and implement. They allow operators to look at information in files. Inquiry requirements might vary from user to user: some users might need to look at data that pertains to their department only, and other users might need to inquire into entire master file records.

When used for inquiry, a display station is not operated continuously. Rather, an operator typically asks a question of the system. Based on the system response, another question might be asked. While the operator reads the displayed information, the system can handle requests from others or can resume processing until the operator asks another question. When an operator finishes inquiring, the display station can be used to do other work.

FILE UPDATE PROGRAMS

Interactive file update programs update master files with transaction file data. How and when the changes occur vary with the type of system design implemented. An effective method of file update that provides immediate update and efficient recovery is called memo updating. Refer to Chapter 4, *Coding Techniques*, for a description of this method.

PROGRAM ATTRIBUTES

Program attributes describe a program's use of display stations or use of resources on System/34.

Attributes that can be specified when a program is compiled are:

- SRT (Single Requestor Terminal). The program allows one requesting display station or SSP-ICF session.
- MRT (Multiple Requestor Terminal). The program allows more than one requesting display station or SSP-ICF session.
- NEP (Never-Ending Program). This attribute can be given to SRT programs and MRT programs. Programs do not wait for nonshared resources that the NEP uses, and the NEP remains active when no requestors are attached to it.

A program can also run without a requestor. This allows a display station to be released from a job step after that step has been initiated if interaction between the display station and the program is not required.

If none of the steps of a job communicate with display stations, the job can be run from the input job queue.

For a description of each of these attributes, refer to *Program Attributes* in Chapter 2.

Usually an application has a mixture of these attributes for its programs. For example, the sample order entry application in Chapter 5 has an SRT program, an MRT program, and a no-requestor-terminal program. The following information provides considerations for choosing program attributes.

If a program is likely to be requested by more than one display station concurrently, consider coding an MRT program. Coding a program as an MRT program avoids resource conflicts that might occur if multiple copies of the program were run concurrently. Also, a single copy of an MRT program usually occupies less storage than two or more copies of the same program coded as an SRT program.

If the program runs when main storage might be overcommitted–the programs that are running do not fit into storage at one time–an MRT program can reduce the swapping that would occur if multiple copies of the same program were run concurrently. Reduced swapping should shorten response times for the display stations.

Finally, only the first requestor of a MRT program causes the program to be initiated. Subsequent requestors should have a shorter sign-on time because their display stations attach to an active program and initiation is not done.

An MRT program might be more complex and use more main storage than the same program coded as an SRT program. If a program will not be requested by more than one operator concurrently and if the initiation time for the program is acceptable, consider coding the program as an SRT program.

If the maximum number of requesting display stations is already attached to an MRT program, the SSP queues a new requesting display station to the program. While the display station waits for its request to be accepted, the display station cannot be used unless the operator presses the Attn key and releases the display station from the MRT procedure. To avoid this situation, you can code the program as an SRT program or increase the maximum number of display stations supported by the program.

If the program must do extensive input/output processing between displays (for example, extensive array processing, multiple printed lines, or ten or more disk accesses), shorter response times are possible when multiple copies of an SRT program are run concurrently.

If a program is requested frequently, is active for more than a few seconds, and uses nonshared resources such as a printer or nonshared disk files, you might want to define the program as never-ending. (Refer to *Never-Ending Programs* in Chapter 2 for a description of these programs.)

An MRT never-ending program with no active requestors will wait for a requestor. This waiting saves program initiation time but will use system resources such as assign/free space in order to remain active.

DISK ACTIVITY FOR LOADING PROGRAMS AND ATTACHING DISPLAY STATIONS TO THEM

The following table shows the number of disk accesses required for loading programs and for attaching display stations to them. Factors affecting the number of disk accesses that are shown in this table are:

- Program attributes
- History file logging
- Read-under-format processing. This technique, a method of overlapping data entry time and program initiation time, is described later in this chapter.
- File status: open or closed
- Number of files used by the program

Program Attribute	History Logging	Read-Under-Format	Files Open	Disk Accesses for Program Load	Disk Accesses for Display Station Attachment
MRT-NEP	Yes	No	N/A	N/A	9
	No	No	N/A	N/A	8
	Yes	Yes	N/A	N/A	3
	No	Yes	N/A	N/A	2
MRT	Yes	No	No	41 + (4 x number of files)	6
	No	No	No	30 + (2 x number of files)	5
	Yes	No	Yes	41 + (2 x number of files)	6
	No	No	Yes	30	5
	Yes	Yes	No	37 + (4 x number of files)	2
	No	Yes	No	26 + (2 x number of files)	1
	Yes	Yes	Yes	37 + (2 x number of files)	2
	No	Yes	Yes	26	1
SRT	Yes	No	No	34 + (4 x number of files)	N/A
	No	No	No	23 + (2 x number of files)	N/A
	Yes	No	Yes	34 + (2 x number of files)	N/A
	No	No	Yes	23	
	Yes	Yes	No	30 + (4 x number of files)	N/A
	No	Yes	No	19 + (2 x number of files)	N/A
	Yes	Yes	Yes	30 + (2 x number of files)	N/A
	No	Yes	Yes	19	N/A

Minimizing Disk Activity to Increase Throughput on the System

There are several things you can do to minimize disk activity and disk processing.

- Call a procedure from within another procedure by using the procedure name instead of a // INCLUDE OCL statement.
- Log entries to the history file only if they are absolutely necessary. Each record logged to the history file requires three to four disk operations.
- Send messages to work station operators by using a // PAUSE OCL statement instead of a //* statement.
- Use the IDELETE command to suppress informational messages that may not be necessary for a particular work station.
- Use tests for the existence of disk, diskette, or library members only when they are necessary. These tests require reading the disk or diskette VTOC and require additional disk activity. You should be aware of this when coding procedures, and should branch around these expressions in your OCL when these existence tests are not needed.

PROGRAM SIZE

If possible, programs should be designed to run in a predetermined region size that allows more than one program to be resident in user storage concurrently. If a program is so large that no other program can be in user storage with it, another program can be swapped in, but these programs cannot execute concurrently.

A program that is larger than the predetermined region size can cause two or more programs or segments of programs to be swapped out each time the large program is swapped in. This swapping can affect performance because of the additional disk activity that is required. Swapping, however, is more efficient than using overlays for a program if the execution of one program cycle results in many overlays. For example, in RPG the number of overlays is not easily controlled by a programmer. In COBOL, Basic Assembler, FORTRAN, BASIC, and WSU, the programmer can control the overlays and therefore might improve performance by using overlays instead of swapping. In addition, BASIC has a status feature in the HELP support that allows the programmer to see how many overlays are occurring for a particular region size and work area partition.

When you predetermine a region size and try to code your programs to fit that region, you might have to adjust the size of the program after you have coded it. If the program is too large, specifying the predetermined region size to execute causes overlays to be built to fit the program into the region.

Each program need not be the same size in order to execute in the same region. If the region size specified for a program is not a multiple of 2 K bytes, the number is rounded up to the next even 2 K byte increment. For example, if you have a 32 K byte user memory, you might design each program so that it executes in a 16 K byte region. Because of the rounding up that was previously described, programs between 14 K bytes and 16 K bytes will require this region for their execution.

Program size can vary with the number of functions and the types of functions used and is therefore difficult to estimate before a program is coded. For example, the following items can affect program size:

- Number of files used.
- File types used. For example, input-only disk files require less storage than update-capable disk files.
- Processing method used. For example, input-only processing of disk files require less storage than update or add capable processing.
- Storage index for indexed files.
- Amount of input processing specified. For example, the number of I specifications in an RPG program can affect program size.
- Record lengths, which can affect the size of input/output areas that are reserved for files.
- Number of output specifications.
- Number and type of calculations. For example, a LOKUP operation in an RPG program done by a subroutine. If you use this operation and a manual search in an RPG program, you might want to use one method instead of two in order to reduce the storage required.
- Number and sizes of constants, data structures, fields, tables, and arrays defined.
- The number of formats on an H specification in an RPG II program. If a number is not specified, 32 is assumed. The number of formats must be at least the number of formats in the load member, even if not all of the formats are used in the program. For each format, an additional 16 bytes is required in your program. For example, if your program uses five formats and you specified five formats on the H specification, 80 bytes (5 x 16) are used for formats. If, for the same program, you used the default of 32 formats on the H specification, 512 bytes would be required. Thus, your program would be 432 bytes larger than necessary.

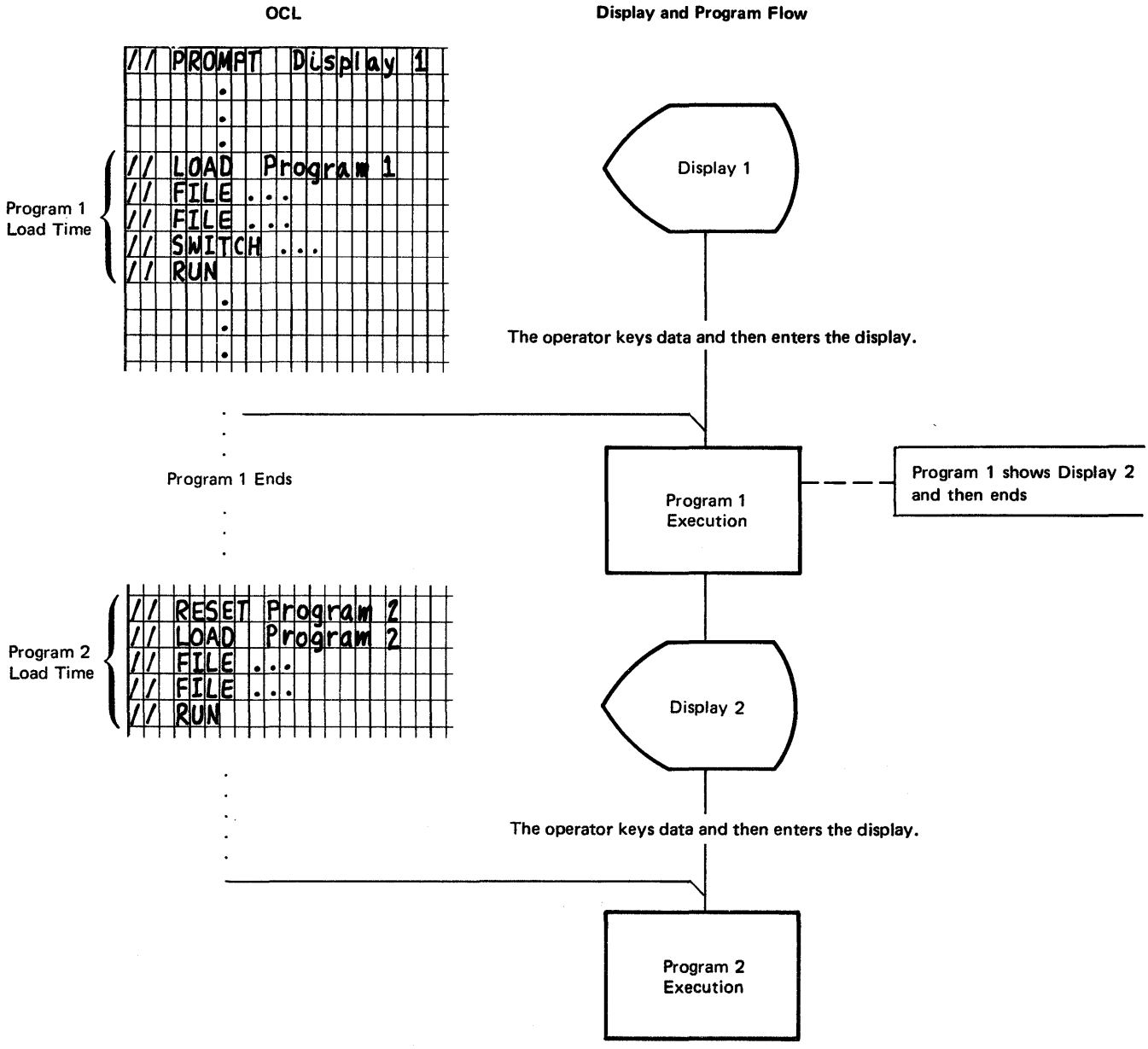
Program size can be adjusted by dividing a program into several job steps and using a technique such as the read-under-format technique to show displays.

READ-UNDER-FORMAT (RUF)

Using a read-under-format technique allows an operator to enter information onto a display while the program that uses the display is initiating. When read-under-format is used, a program or a procedure displays the format, and the program called next in the procedure reads it. The format is displayed by a program or a PROMPT OCL statement with PDATA-YES specified. If an SRT program displays the format, it then goes to end of job. An MRT program displaying the format releases the display station. While the program is being initiated, the operator enters information for the display. When the operator presses the Enter key, the input from the display is sent to the second program.

This technique can be used with all types of programs, including never-ending programs. Read-under-format processing decreases program size because each program might handle a few formats. This technique might increase processing time because of the extra time the system spends initiating a program.

The following example shows a read-under-format technique that uses two displays and two programs. The PROMPT OCL statement is used to show Display 1. While the operator enters information on that display, Program 1 is being loaded. When the operator enters Display 1, his input is sent to Program 1. Before Program 1 ends, it shows Display 2. The operator can enter information on this display while Program 2 is being loaded. When the operator enters Display 2, his input is sent to Program 2.



The sample application described in Chapter 5 also uses a read-under-format technique.

DISPLAY STATION LOCAL DATA AREA

A 256-byte local data area exists on disk for each command display station on the System/34. This area may be used to pass information between programs and procedures. This area is initialized to blanks at the start of a session. RPG II, WSU, SDA, SMF and 3270 emulation use part of the display station local data area to control their execution. Therefore, any user data in those bytes is destroyed when one of these programs is run. The use of the local data area by these programs is as follows:

Program	Bytes Used
ICFVERIFY	1 through 4
SDA	1 through 104
RPG II	201 through 256
SMF	220 through 256
3270 Emulation	230 through 256
WSU	238 through 256
HELP	249 through 256

The LOCAL OCL statement may be used to put data from a procedure into the local data area. The ?L'offset,length'? substitution expression may be used to test or extract data from the local data area in a procedure. Both COBOL and RPG II have subroutines available to read and update the local data area for any attached display station. In BASIC, the local data area can be opened with the OPEN statement and is then available to read and update.

The local data area becomes resident in main storage the first time a LOCAL OCL statement or LOCAL substitution expression is encountered within a job step. Each subsequent LOCAL OCL statement processed in a job step updates the main storage resident local data area and the data area that resides on disk. Each subsequent LOCAL substitution expression in the job step accesses only the local data area in storage. The local data area is resident in storage for a particular job step only until a // RUN OCL statement is encountered and processed. The local data area can be reestablished for the next job step if necessary.

The local data area, when resident in main storage, uses the assign/free area of the nucleus.

The following OCL statements in a procedure called PROCA would prompt the operator to enter a starting invoice number. It would store this number in the first six bytes of the local data area. PROC5 would execute using and updating the first six bytes of the local data area, which contain the invoice number. When control returns to PROCA, the updated invoice number would be displayed to the operator and the procedure would pause until a 0 response is entered.

```
// * 'ENTER STARTING INVOICE NUMBER 6 DIGITS'
// LOCAL OFFSET-1,DATA-'?1R?'
PROC5
// PAUSE ' ENDING INVOICE WAS ?L'1,6'?'
```

Note: If a // * statement, rather than the PAUSE statement, is used to display the operator message, processing of the procedure continues and the message might appear on the screen only momentarily. For that reason, if a // * statement is used, you might want to follow it with a PAUSE statement.

EXTERNAL INDICATORS

System/34 has a set of eight external indicators (switches) available for each display station. These switches are available to the COBOL programmer as switches UPSI-0 through UPSI-7 and to the RPG II and WSU user as indicators U1 through U8.

Subroutines are provided for retrieving and updating the external switches of any attached, command-capable display station in an RPG MRT program or in an SRT or MRT program coded with the subset ANS COBOL PRPQ (X3.23-1968). Retrieving and updating these switches is automatically done for RPG SRT programs, WSU programs, and programs coded with the ANS COBOL Program Product (X3.23-1974). The switches are available to BASIC programs through the UPSI$ intrinsic function.

System/34 OCL also has the ability to test and set these switches. Thus, the execution flow of various OCL statements in a procedure can be controlled by the settings of various switches.

If UPSI-YES is specified on the // PROMPT statement, the current setting of the eight UPSI switches are made available to the screen format as indicators 91 through 98. A return code that indicates which command or function key was used can be accessed by a ?CD? substitution parameter. For more information on using the ?CD? substitution parameter with indicators, refer to *Using the Prompt OCL Statement* in Chapter 4.

The following example uses the SWITCH OCL statement to set switch 1 to an off status and switch 2 to an on status; switches 3-8 are unchanged. PROC1 is called and executed. If switch 1 is on when control is returned from PROC1, then PROC2 is called; if switch 1 is off, the ELSE statement causes PROC3 to be executed. The last SWITCH statement sets all eight switches to an off status. The switches are also set to an off status when a display station session is initiated.

```
// SWITCH 01XXXXXX
PROC1
// IF SWITCH1-1 PROC2
// ELSE PROC3
// SWITCH 00000000
```

Note: A separate copy of the switch settings is kept for each requestor of an MRT program. When a requestor gains control of the program, the switches are automatically set to the values stored for that requestor.

Data Processing Security and Accuracy

Data processing security involves protecting both data and the equipment needed to process the data. The major emphasis of data processing security is to prevent the unauthorized release, modification, or destruction of your information or data processing equipment. Data processing security is especially important because the System/34 allows many activities to occur at the same time, often away from as well as at the central computer site. There are two important areas that make up data processing security: physical security and data security.

PHYSICAL SECURITY

Place your computer in a safe location. There are several factors to consider.

Physical Location

You should not place your computer below ground level, as in the basement. Backed-up sewer lines, broken water mains, and floods can occur.

Place the computer away from outside walls or windows. If you must place the computer along an outside wall or window make sure the wall or window is strong enough to protect your computer from damaging winds, hail, or other conditions accompanying severe weather.

Limited Access to the Computer

Only people who need to use the computer to do their work should be allowed to use the computer. If you have the computer in a special room, you may want to limit access to the computer room through the use of special locks or special doors.

Fire Protection

Place your computer in a building that is as fireproof as possible. Good housekeeping is vital to maintain a fireproof environment.

Other things you can consider:

- Place some fire extinguishers near the computer and make sure the appropriate people are trained in their use.
- Keep a smoke detector near or in the computer room.
- Install appropriate sprinklers to put out a fire.

  Because sprinklers can use water that can damage your computer, make sure your sprinklers use chemicals that will not damage the System/34.

DATA SECURITY

Limited Data Access

Limiting access to your data protects it from being read or disclosed to people who are not authorized to use it. The System/34 provides some security features to help you safeguard your data. These features include:

- Password security
- Badge security
- File and library security
- Menu security

For more information about System/34 security features, refer to *System-Provided Security* in Chapter 2.

Data Accuracy

You can better safeguard the accuracy of your data if you use certain data controls to help you make sure your data is as accurate, reliable, and complete as possible. There are three basic types of data controls:

- Input controls
- Processing controls
- Output controls

Input Controls

The following steps will help to ensure that your input data is correct.

Input Verification: Check fields on the input record to see if these fields are correct. Some businesses have personnel check all input documents and records before entering them into the computer.

It is necessary to make sure not only that input records are processed correctly but that all input documents cannot be lost and the loss go undetected. You should have some way of recording what input was entered into the computer.

Processing Controls

Processing controls are those routines that are written into a program to ensure the program is processing data correctly.

Some common processing controls are:

- Record counts: This control counts how many input records were processed; it can determine if any records were lost during processing.
- Sequence checking: This control checks whether records have been sorted properly.
- Audit trails: An audit trail records what work was done on the computer and the order in which the work was done. An audit trail should indicate that the computer is doing the work correctly. Additionally, the audit trail should provide information to identify any errors and their causes.

Output Controls

These controls report the results of the processing done by the computer. These controls when combined with input controls can be especially effective in checking such items as output totals compared to input totals. Some effective output controls are:

Output counts: Count of records either processed or written as output.

Program messages: Messages issued by the program when data errors occur (for example, a message issued to the console operator when a four-digit control number is blank and the control number is used as a key).

BACKUP AND RECOVERY CONSIDERATIONS

Because data (programs and files) can be damaged or destroyed by incorrect modification, a system failure, operator errors, or a natural disaster such as a fire, keeping backup copies of vital information is recommended. Backup procedures typically involve copying the vital information kept on the system and then storing the copy in a safe location. For example, data can be copied to diskettes, dated, labeled, and stored in a fireproof safe. Because a disaster such as a fire could destroy the onsite backup copy, a good practice is to store another set of backup diskettes offsite.

Loss of data could be disastrous to most businesses using data processing. Thus, a standard, well-documented, backup procedure should be established and used regularly. Typically, master files and all files related to the master files are saved at the same time. For example, if a customer master file contains an accounts receivable sum for each customer, this file and the accounts receivable open item file are saved and restored together.

New data such as batches of transaction records can be copied to disk or diskette after they have been entered and edited. These saved transactions can be used during recovery procedures to make the master files current.

Recovery is a series of steps that an operator follows or procedures that an operator runs to restore data on the system. Following recovery, programs and files are returned to the status that they had just before the error, failure, or disaster occurred.

Recovery procedures can require removing all or some master files, restoring backed up master files, and reexecuting those procedures that updated files to repost transactions in the order that they were originally processed.

Programs and procedures can be designed to restore and recover all files, inform the operators about the last items correctly processed, and allow operations to continue from that point. This effort might involve using additional fields in records and using additional calculations in programs. Also, new files, programs, and procedures might be needed, particularly for recovery in a work station environment. The planning and programming effort might not seem costly in light of the potential results of inadequate backup and recovery procedures. Typically, businesses that are most dependent on their data processing system require the shortest recovery times and thus should develop the most elaborate backup and recovery procedures. Regardless of their complexity, backup and recovery procedures should be well-documented so that all operators use them correctly.

The following information describes three methods of backup and recovery. The first method requires the least design and programming effort, but probably requires the longest recovery time because transaction batches are not saved. The second method requires more planning and programming, but reduces the amount of recovery time required because reentering the transactions is not necessary. The third method requires the most planning and programming but provides the quickest way to recover data because the operator's involvement is minimized.

Method 1

This method requires the operator to periodically save master files and files that the application updates in order to establish a point from which to recover (restart) the application. For example, at the end of each day after all transactions have been posted, the operator might execute a procedure that contains SAVE commands to back up all master files and their related files on diskette.

Operators should keep a log of the work they do on the system. This manually kept log must be accurate if it is to be relied upon during recovery. One method of keeping a log is to use the following sample run sheet.

RUN SHEET

Work Station ID _____ Date _____ Page _____

Menu, Item, Command, or Procedure Name	Operator's Initials	Start Time	Stop Time	OK	Comments, Halts, Messages

Another method of obtaining a log of work done on the system is to print the history file.

This recovery method consists of the operator (1) deleting files from disk, (2) restoring the backup copies from diskettes to establish a point from which to recover, and (3) reprocessing all transactions that have been entered since the last backup was done.

All of the work done since the last backup must be redone. Because they are not saved, transaction batches must be reentered. This method might be adequate for a business that processes low volumes of data and that frequently backs up its data.

Method 2

This method requires the operator to (1) periodically save the master files and their related files and (2) save batches of transactions at logical breakpoints in the application. For example, at the end of each day after all transactions have been posted, the operator executes a procedure that contains SAVE commands to back up master files and their related files on diskettes. As part of the transaction-posting procedure run during normal processing, a batch of transactions is saved on diskette and deleted from disk. The operator labels the diskettes that contain the transactions so that he knows the sequence in which the batches have been saved. Also, the operator lists the names of the procedures in the order that he runs them.

The recovery method requires the operator to (1) delete the files from disk, (2) restore the backup copies from diskettes to establish a point from which to recover, and (3) reprocess the application's procedures in their original order using the saved copies of the transaction batches. The operator uses the information he has labeled on the diskettes to ensure that the transaction batches are restored in the correct order.

This method eliminates the rekeying of transactions that was required in the previous method.

Method 3

This method requires code to be included in an application's procedures to do the following things:

- Periodically back up the master files and their related files.
- Automatically back up batches of transactions on disk or diskettes at logical breakpoints in the application.
- Assign names and sequence numbers to these batches of transactions.
- Keep a history of all procedures executed by the operator following the last backup. This history is kept in a control file.
- Provide a common recovery procedure.

The recovery method consists of the operator running the common recovery procedure, which lists the control file and restores the files. The operator uses the listing of the control file to rerun the application's procedures in their original order. The common recovery procedure could prompt the operator to insert the proper backup diskettes in the correct sequence.

This recovery method uses a program-generated control log, which is more accurate than a manually-kept log. Because unnecessary procedures such as reprinting statements or reports could be skipped during recovery, this method provides the quickest recovery of the three methods. This method is similar to the backup and recovery procedures used in some IBM Licensed Application Programs.

The following example shows one method of restoring a particular file from a diskette in a magazine.

Step	Procedure Command	Action
1	CATALOG	Determine the file that you want to restore.
2	RESTORE	Restore file to disk from diskette files kept as backup.
3	CATALOG(optional)	Verify that the file has been restored.

Procedure Command Examples

1. Print a list of all files on each diskette in a magazine.
   CATALOG ALL,I1,M1.01,NOAUTO

2. Restore a file named ORDITEM located on diskette 07 in magazine drive 01.
   RESTORE ORDITEM,,,,M1.07

History File

The history file is an area that is located in the system area on disk (#SYSHIST) and occupies a minimum of 120 sectors and a maximum of 9960 sectors. The size of the history file is specified during the configuration process. The history file is an important tool you can use to review events that have occurred on your system. Recorded in the history file are:

- All OCL statements, utility control statements, control commands, and procedures executed by the SSP.
- All messages displayed at the display station.
- All operator responses to messages and prompts.
- Work stations being used.
- Operator's user ID.
- Job name in the form WWHHMMSS; WW is the work station ID and HHMMSS is the time the job began running. Occasionally, there will be all asterisks (*) in either the operator user ID or the job name. This indicates that the entry recorded was generated by the system. Double quotes ('') indicate that the line is continued from the previous entry.
- The time the entry was recorded in the history file.

In addition, there are entries in the history file designated as *E entries. There are two types of *E entries: *EJ, which indicates the end-of-job recording, and *EP which indicates the end of a spool print job. The following information items are shown by the *E entries:

- Starting and ending time of the job
- Date the job was run
- Amount of elapsed time it took to run the job
- Operator's user ID
- Work station or printer ID
- Name of the procedure that began the job or created the print job
- Whether the job was an MRT program
- Two-character completion code for the *EP entries
  - NC: Normal completion
  - CP: The print job was canceled by either the console or subconsole operator
  - SP: The print job has been stopped by either the console or subconsole operator

The following three printouts are samples of a listing of the history file entries.

```
HISTORY FILE DISPLAY              WORKSTATION - X1    USER - RON         DATE  07/24/80     TIME  14.01     SUBCONSOLE - PAGE-    1

STATEMENT                        HISTORY SYSTEM                                    WKSTN  USER      JOB NAME     TIME
---------                                                                          -----  ----      --------     ----
  RENAME TEMP1,WFR7V066                                                             W2    RON       W2143606   14.36.07
   RENAME PROCEDURE EXECUTING                                                       ''    ''          ''       14.36.08
  HELP SAVE                                                                         ''    ''        W2143617   14.36.17
  SAVE WFR7V066,999,,WSUWSU,,,,,,                                                   ''    ''          ''       14.36.30
   SAVE PROCEDURE EXECUTING                                                         ''    ''          ''       14.36.32
  SEU SAVEFILE,P,,,WSUFT                                                            ''    ''        W2143700   14.37.01
  SEU PROCEDURE EXECUTING                                                           ''    ''          ''       14.37.02
  // LOAD $SFGR                                                                    X1    JMG       X1143800   14.38.01
  // RUN                                                                           ''    ''          ''       14.38.05
  // LOADMBR NAME-TOM                                                             ''    ''          ''       14.38.14
  // INOUT INLIB-JMGLIB,OUTLIB-JMGLIB                                             ''    ''          ''       14.38.27
  // UPDATE SOURCE-TOM                                                            ''    ''          ''       14.38.49
  SAVEFILE 066,CUSMF001,ITEMF002,,TRANF018,58,25                                    W2    RON       W2143850   14.38.50
   DISPLAY PROCEDURE EXECUTING                                                      ''    ''          ''       14.38.52
  // END                                                                           X1    JMG       X1143800   14.38.56
  SYS-1598  OPTIONS (  23) COCI                                                     W2    RON       W2143850   14.38.59
  TRANF018--THIS FILE IS NOT ON DISK                                                ''    ''          ''       14.39.00
  SYS-5019  OPTIONS (   3) SFDE                                                     X1    JMG       X1143800   14.39.01
  TERMINAL ERRORS IN $SFGR INPUT SPECIFICATIONS                                     ''    ''          ''       14.39.01
  3                                                                                 ''    ''          ''       14.39.05
  3                                                                                 W2    RON       W2143850   14.39.19
  SDA                                                                               X1    JMG       X1143926   14.39.27
  SEU SAVEFILE,P,,,WSUFT                                                            W2    RON       W2143932   14.39.32
  SEU PROCEDURE EXECUTING                                                           ''    ''          ''       14.39.34
  SEU-0549  OPTIONS (  2 ) SEEJ                                                     ''    ''          ''       14.40.36
  NOT ENOUGH ROOM IN LIBRARY TO REPLACE MEMBER                                      ''    ''          ''       14.40.36
  2                                                                                 ''    ''          ''       14.40.41
  CONDENSE WSUFT                                                                    ''    ''        W2144046   14.40.46
   CONDENSE PROCEDURE EXECUTING                                                     ''    ''          ''       14.40.47
  SYS-2582  OPTIONS ( 123) MARC                                                     ''    ''          ''       14.40.49
  WSUFT   --THIS LIBRARY NOT COMPRESSED, BEING USED...                              ''    ''          ''       14.40.49
  3                                                                                 ''    ''          ''       14.40.52
  OFF                                                                               ''    ********  ********   14.41.02
  OFF                                                                               R2    ********  ********   14.41.16
  SEU PROCEDURE EXECUTING                                                           X1    JMG       X1143926   14.41.24
          LDK                                                                       W3    LDK       ********   14.41.28
  // LIBRARY NAME-WSUFT                                                            W1    RON       W1144131   14.41.31
  CONDENSE WSUFT                                                                    W3    LDK       W3144138   14.41.38
   CONDENSE PROCEDURE EXECUTING                                                     ''    ''          ''       14.41.39
  // LIBRARY NAME-$LIBRARY                                                         W1    RON       W1144140   14.41.41
          CJR           CMENU WSUFT                                                 W2    CJR       ********   14.41.55
  OFF                                                                               W3    ********  ********   14.41.59
  SEU SAVEFILE,P,,,WSUFT                                                            W2    CJR       W2144200   14.42.01
  SEU PROCEDURE EXECUTING                                                           ''    ''          ''       14.42.02
  SEU-0545  OPTIONS (01 3) SE                                                       ''    ''          ''       14.42.07
  WORK FILE ALREADY EXISTS--IS THIS A RECOVERY RUN?...                              ''    ''          ''       14.42.07
  0                                                                                 ''    ''          ''       14.42.13
  WSU WSUFT079,WSUFT,,REPLACE                                                       ''    ''        W2144551   14.45.51
   WSU PROCEDURE EXECUTING                                                          ''    ''          ''       14.45.53
          RON           COST  WSUFT                                                 W3    RON       ********   14.46.14
          RON           .     WSUFT                                                 W4    ********  ********   14.46.35
```

```
HISTORY FILE DISPLAY              WORKSTATION - X1    USER - RON        DATE  07/24/80    TIME  14.05    SUBCONSOLE - PAGE-    1

                                HISTORY ALL,,CURRENT,TEXTONLY
    HISTORY NOLIST,RESET,,,,X2
      // LIBRARY NAME-0
      // MEMBER USER1-$$MSG2
      // * 2076                                  @@206
   HISTORY PROCEDURE EXECUTING
      // LOAD $HIST
      // RUN
      // DISPLAY NOLIST,RESET,,,
      //  WORKSTN-X2,
      // SYSTEM-NO
      // GOTO END
      // END
*EJ  07.28.07  07.29.13  00.01.06  07/24/80  RON       X1   PUBSHISR

  * W1,RON       NO OPERATOR COVERAGE FROM 11:30 TO 13:30!!
          RON          COST  PUBSLIB
  HELP HISTORY
      // LOAD $HELP
      // RUN
      // * 'HISTORY SYSTEM,VIEWED,NORESET,TOTAL,CONTROLS,,,,,'
  HISTORY SYSTEM,VIEWED,NORESET,TOTAL,CONTROLS,,,,,
    HISTORY SYSTEM,VIEWED,NORESET,TOTAL,CONTROLS,,,,,
      // LIBRARY NAME-0
      // MEMBER USER1-$$MSG2
      // * 2076                                  @@206
   HISTORY PROCEDURE EXECUTING
      // LOAD $HIST
      // RUN
      //  GOTO SYSOVFL
      // DISPLAY VIEWED,NORESET,TOTAL,CONTROLS,
      //  SYSTEM-YES
      // END
  SYS-8423 SRT CANCELED BY INQUIRY OPTION 3 AT X1
*EJ  13.57.23  13.59.09  00.01.46  07/24/80  RON       X1   HISTORY
```

Operator entry

Generated OCL

System display

Generated OCL

*EJ entry

```
  HELP HISTORY
      // LOAD $HELP
      // RUN
      // * 'HISTORY SYSTEM,VIEWED,NORESET,TOTAL,CONTROLS,,,,,'
  HISTORY SYSTEM,VIEWED,NORESET,TOTAL,CONTROLS,,,,,
    HISTORY SYSTEM,VIEWED,NORESET,TOTAL,CONTROLS,,,,,
      // LIBRARY NAME-0
      // MEMBER USER1-$$MSG2
      // * 2076                                  @@206
   HISTORY PROCEDURE EXECUTING
      // LOAD $HIST
      // RUN
      //  GOTO SYSOVFL
      // DISPLAY VIEWED,NORESET,TOTAL,CONTROLS,
      //  SYSTEM-YES
      // END
  SYS-8423 SRT CANCELED BY INQUIRY OPTION 3 AT X1
```

```
HISTORY FILE DISPLAY              WORKSTATION - X2    USER - DRF        DATE  09/08/80    TIME  11.44    SUBCONSOLE - PAGE-    1

STATEMENT                       HISTORY ALL,,,EONLY                                    WKSTN  USER     JOB NAME     TIME
*EJ  11.35.49  11.36.38  00.00.49  09/08/80  DRF       X2   SEU                          X2   DRF      X2113549   11.36.38

     1 ENTRIES DISPLAYED    ..
```

During system configuration you can also specify whether you want an automatic wraparound capability for the history file. If you specify the wraparound capability, once the file becomes full of entries, each new entry to the file causes the oldest entry to be deleted from the file. When the history file is listed, the oldest entry is displayed or printed first on the output device you specify.

You can also specify an overflow file for the history file during configuration by taking the override options on the IPL sign-on screen. The overflow file helps you avoid losing entries once the history file is full. The overflow file contains one to eight segments, and each segment is the same size as the history file. If fewer than 25 sectors remain in the history file, the system issues an informational message to the system console and copies the contents of the history file into an overflow file segment. Entries in the history file and the overflow file at this time are identical.

To print or display the contents of the overflow file, use the HISTORY procedure. You should reset the overflow file when all entries in a segment have been displayed, so that the system can reuse that segment for another copy of the history file. If both the overflow segment and the history file are full of entries, the system does an automatic wraparound of entries until the overflow file is emptied and reallocated by the RESET parameter of the HISTORY procedure. Be very careful when using the RESET parameter because it is possible to reset one display station's history entries from another display station.

Note: You cannot request all history entries for active jobs in the system because the results would be unpredictable and the system performance would be slowed down.

HISTCRT Procedure

The HISTCRT procedure allows you to selectively view the contents of the history file. Using the HISTCRT procedure, you can page forward and backwards through entries in the history file as well as view all entries, beginning with the most current entry, that match criteria you have defined, such as job name or work station ID.

For more information about the HISTCRT procedure, refer to the *SSP Reference Manual.*

CONSIDERATIONS FOR REMOTE WORK STATIONS

Program performances can be significantly affected by whether a local work station or a remote work station is used. Local work stations communicate with the System/34 at a rate of 1 000 000 bps (bits per second). Remote work stations communicate at a rate of 1200 bps to 9600 bps.

Transmission of a format that contains 4000 characters, the maximum number of characters in a format for a 1920-character display, takes at least 27 seconds on a 1200 bps line and at least 4 seconds on a 9600 bps line. These time estimates apply to optimum line conditions. Because of this relatively slow transmission rate, remote work stations can be a bottle-neck for system activity. If remote work stations seem to cause poor performance, two alternatives are available that might significantly improve performance:

- Reducing the amount of data that is transmitted over the line
- Increasing the line speed

Of these two alternatives, reducing the amount of data transferred is the only one directly controlled by programming techniques. In most cases, you can reduce the size of the data stream by:

- Sending only the minimum amount of data required by an operator to efficiently use the application. If both experienced and inexperienced operators use the application, consider sending only the information required by the experienced operator and allowing the inexperienced operator to request additional information.
- Avoiding the retransmission of data and prompts.
- Using multiple formats, one for constant information such as headings and others for variable data.
- Using the erase input operation when practical rather than rewriting input fields.
- Using variable start line numbers or rewriting the format.
- Avoiding the use of the display screen to pass information from one job step to another. Instead, consider using a disk file, the local data area, or data structures or arrays to pass the information.
- Using put override for RPG II, WSERROR for COBOL, REWRITE for BASIC, or IMSG for WSU to display error messages. These techniques avoid retransmission of data when errors occur.
- Avoiding out-of-sequence fields on formats. Approximately two additional characters of data are transmitted for each out-of-sequence field.
- Using PRINT NEWPAGE and TAB for BASIC rather than letting lines roll up on the display.

- Spooling output data to remote printers. When spooled output is sent to a printer, blanks in the data stream are compressed. When output that is not spooled is sent to a printer, all characters, including blanks, are transmitted.

- Avoiding functions that require the contents of the display screen to be saved and later restored. Such functions require the transfer of approximately 3.5 K bytes of information from and to the display station. The following functions cause the entire display screen to be saved and can degrade performance of a remote display station:
  - Using the Attn key to interrupt the job.
  - COBOL programs that use work station support and DISPLAY and ACCEPT.
  - Changing the mode of a WSU program.
  - Using the WSU menu display.
  - Receiving informational system messages or messages sent via // * statements when a format is already displayed. You can suppress these messages using the IDELETE control command.
  - Using the PAUSE OCL statement.
  - Using PAUSE or TRACE in BASIC when the work station file is open.
  - Interrupting and resuming BASIC programs when the work station file is open.

In addition to the time required for transmitting data, the time required to reverse the direction of transmission can also affect performance. Because the System/34 communication facility transmits in one direction at a time, time is required for the turnaround from receiving to transmitting or from transmitting to receiving.

A 1/4 second turnaround time is not unusual. When multiple formats are displayed before the program reads from the display screen, suppressing input on all but the last format reduces the number of line turnarounds. (Suppress input is specified in columns 35 and 36 of the S specification.) If input is not suppressed, several line turnarounds are required after each format is displayed.

Another important factor that affects the performance of a system and its remote work stations is the size of the work station buffer, which is sometimes referred to as work station queue space or WSQS. All output to remote work stations must pass through the work station buffer. If the buffer is too small, a program may have to wait for a long time before buffer space is available for an output operation. For information about how work station data management uses the work station buffer, see *The Work Station Buffer* in Chapter 2.

Chapter 4. Coding Techniques

This chapter presents coding techniques that can help you program more efficiently:

- Memo updating
- Program communication with the local data area
- Using the PROMPT OCL statement
- Protecting records from concurrent updates in an MRT program
- Protecting records from concurrent updates by multiple MRT programs
- Using the local data area to increase the Sort program's flexibility
- Using data structures for multiple line displays
- Accessing a command key or a function control key in an RPG II program

All these techniques might not apply to your situation. For those techniques you use, you might find variations of your own that tailor the technique to your particular job.

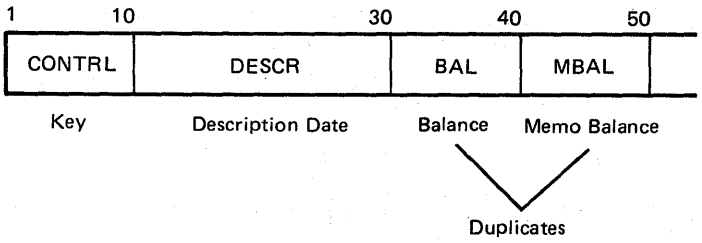
MEMO UPDATING

An advantage of a work station environment can be that operators always have access to up-to-date information in the master files. For example, suppose an operator enters a transaction that reduces the on-hand quantity of an item in an inventory master file. If an inquiry is made by another operator for the on-hand quantity of that item, he can see the value that reflects the previous adjustment made to it.

Allowing interactive updates to files should be done carefully because recovery from a system or program failure can be difficult if you do not know which updates are reflected in the file and which updates need repeating.

Memo updating is a technique that allows interactive updates to your master files and provides batch processing to check that the updates have been applied correctly.

For this technique, master file records must allow duplicate fields for those fields that can be updated interactively. For example, memo balance (MBAL) could reflect interactive updates, and balance (BAL) could be used for batch processing.



Initially (for example, at the beginning of the day), these two fields should be equal. The transactions made during the day are applied only to the memo balance field.

The following RPG input specifications and output specifications could be used for the master file by interactive data entry and inquiry programs. Notice that these specifications ignore the balance field. The memo balance field should always reflect the current balance.

I

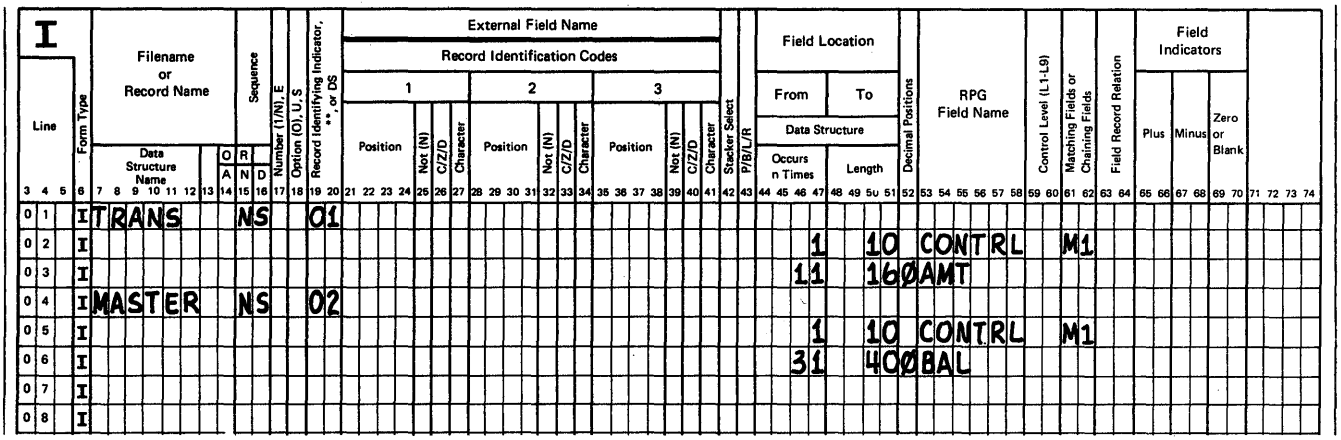
Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Data Structure: Occurs n Times	Data Structure: Length	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
01	I	MASTER	NS			01																										
02	I																				1	10				CONTRL						
03	I																				11	30				DESCR						
04	I																				41	50				ØMBAL						
05	I																															
06	I																															

O

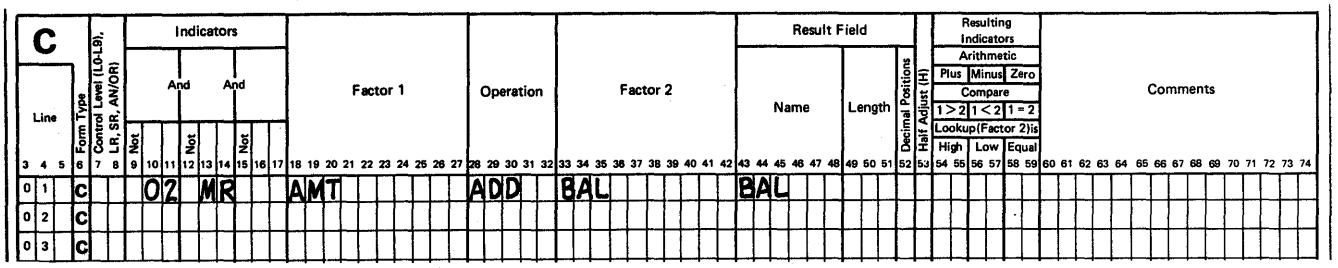
Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space: Skip #/Fetch (F)	Space: Before	Space: After	Skip: Before	Skip: After	Output Indicators: And	Output Indicators: And	Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R	Commas	Zero Balances to Print	No Sign	CR	–	X = Remove Plus Sign; Y = Date Field Edit; Z = Zero Suppress	5 - 9 = User Defined	Constant or Edit Word
01	O	MASTER	D						01														
02	O										MBAL			50									
03	O																						
04	O																						

Commas	Zero Balances to Print	No Sign	CR	–
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

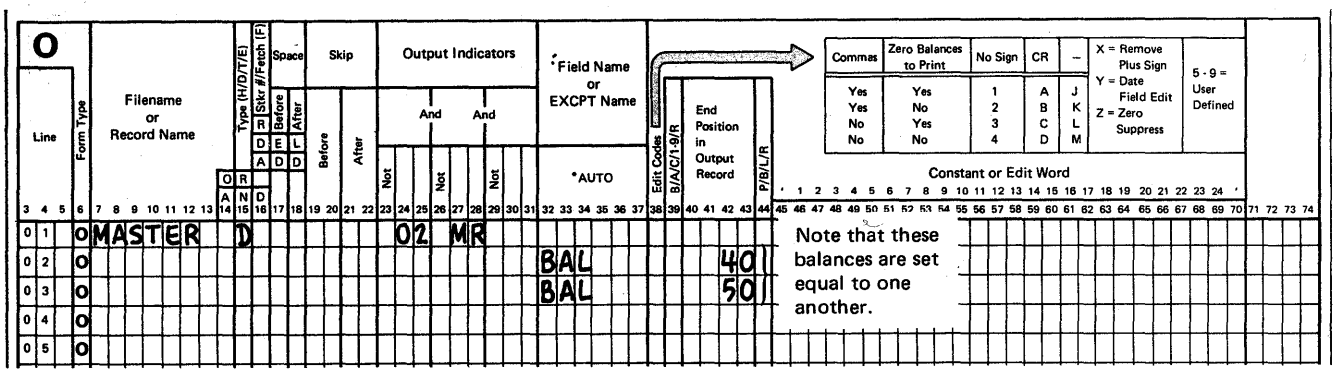
Later (for example, at the end of the day), the transaction file is processed by a batch edit program. The transactions are posted to the balance fields in the master file by a batch update program, as the following segment of the program shows:



I	Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	To	Data Structure: Occurs n Times	Length	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Minus	Zero or Blank
	01	I	TRANS		NS		01																										
	02	I																					1	10		CONTRL		M1				
	03	I																					11	16		ØAMT						
	04	I	MASTER		NS		02																										
	05	I																					1	10		CONTRL		M1				
	06	I																					31	40		ØBAL						
	07	I																															
	08	I																															



C	Line	Form Type	Control Level (L0-L9), LR, SR, AN/OR	Indicators: And	And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Compare: 1>2	1<2	1=2	Lookup(Factor 2)is: High	Low	Equal	Comments
	01	C		02 MR		AMT	ADD	BAL	BAL													
	02	C																				
	03	C																				



O	Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Stkr # / Fetch (F)	Space: Before	After	Skip: Before	After	Output Indicators: And	And	*Field Name or EXCPT Name	Edit Codes	B/A/C/1-9/R	End Position in Output Record	P/B/L/R
	01	O	MASTER	D						02 MR						
	02	O										BAL			40	
	03	O										BAL			50	
	04	O														
	05	O														

Commas	Zero Balances to Print	No Sign	CR	–
Yes	Yes	1	A	J
Yes	No	2	B	K
No	Yes	3	C	L
No	No	4	D	M

X = Remove Plus Sign
Y = Date Field Edit
Z = Zero Suppress

5-9 = User Defined

Constant or Edit Word

Note that these balances are set equal to one another.

Backup of the transaction files is required each day (or for each batch); backup of the master files is required periodically. Recovery can be done by reloading the master files and processing all subsequent transactions. The current transaction file should be intact after the system IPL File Rebuild function is run. After IPL File Rebuild runs, the master file does not reflect the current transactions. To bring the memo balance field to its current value, run a program that updates the memo balance field with the transactions. Then, after the memo balance field has been updated, all current activity has been accounted for, and normal operations can continue.

A variation of the memo updating technique could be to set the memo balance field to zero at the start of the day rather than to the value of the balance field. As for the previous method, interactive updates would be made only to the memo balance field.

The memo balance field would reflect the day's activity for that item. If no transactions for the item occurred, the memo balance would remain zero. In order to determine the current balance, an inquiry program would have to add or subtract the memo balance from the balance in the master file.

PROGRAM COMMUNICATION WITH THE LOCAL DATA AREA

Programs can communicate with the local data area (LDA) via data structures. For a discussion of data structures, see the *System/34 RPG II Reference Manual*. A U data structure (U in column 18) defines the LDA to the program. This data structure (UDS) can be subdivided. For example:

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094-4 UM/050*
Printed in U.S.A.

Program		Keying Instruction	Graphic								Card Electro Number	Page 1 2 of	Program Identification 75 76 77 78 79 80
Programmer	Date		Key										

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, ** or DS	External Field Name / Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
	I																													
01	I	* USE OF THE LDA																												
02	I	DSWRK				UDS																								
03	I																				1	25		AREA						
04	I																				1	6	0	BEGCK						
05	I																				7	12	0	PYDATE						
06	I																				13	21	2	TOTPAY						
07	I																													
08	I																													

Note: Although a filename (DSWRK) is specified in the example, this entry is optional for data structures used as a display station local data area.

The fields can represent whatever the programmer defines and can be used as factor 1, factor 2, or results in calculations, or the fields may be used as input or output fields with input/output operations.

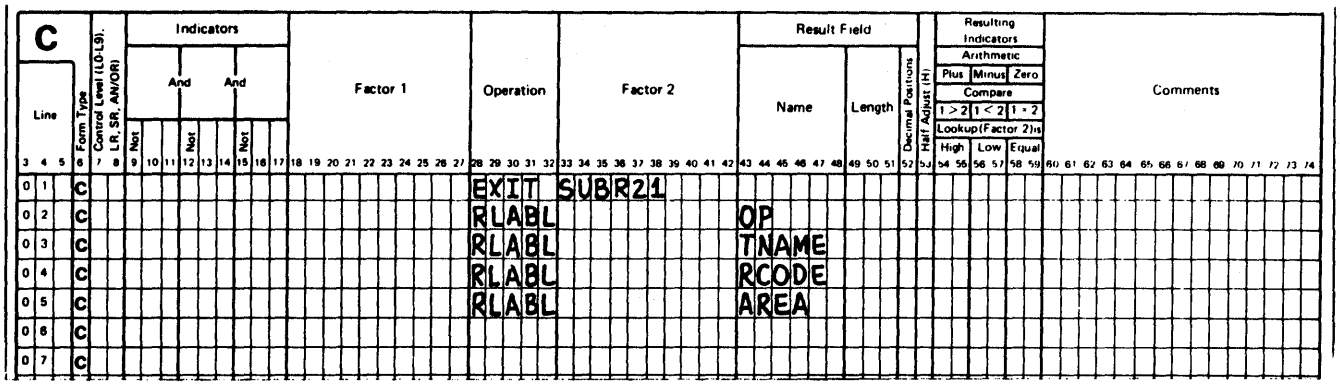
For SRT programs, a routine is provided to load the LDA into the data structure before the first RPG II cycle and to copy it back to the LDA following the last RPG II cycle. This routine is generated if the programmer specifies a U in position 18 of the DS input control record.

A data structure can also be used with an MRT program. However, the only LDA loaded into the data structure is that of the first program requestor. When the program ends, no LDA is updated.

When an inquiry is requested or when a job is placed on the input job queue, a copy of the submitting display station's LDA and user switches is saved. The LDA copy is updated when the program ends so that the LDA can be tested by subsequent programs and OCL in the same job. At end of job, this LDA copy is not saved.

A subroutine, which is mainly for MRT programs, can be used to access the LDA of any display station that is attached to the MRT program. This subroutine, SUBR21, allows the programmer to read or write an LDA, which is identified by the display station ID. SUBR21 requires the two-character display station ID and an indication whether to read or write all 256 bytes of the specified LDA.

The following RPG calculation lines are used for executing SUBR21 from an RPG program:



C	Line	Form Type	Control Level (L0-L9), LR, SR, AN/OR	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Result Field: Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Minus	Zero	Resulting Indicators: Compare: 1>2	1<2	1=2	Resulting Indicators: Lookup(Factor 2): High	Low	Equal	Comments
	01	C					EXIT	SUBR21														
	02	C					RLABL		OP													
	03	C					RLABL		TNAME													
	04	C					RLABL		RCODE													
	05	C					RLABL		AREA													
	06	C																				
	07	C																				

The meanings of the OP, TNAME, RCODE, and AREA fields are as follows:

OP — One-character field that indicates the operation code. The RPG program places this code in the field.

I = input: read from the LDA
O = output: write to the LDA

TNAME — Two-character field that contains the display station ID. If TNAME is the same label used with the ID field of the work station file, then SUBR21 is addressing the LDA for the work station currently being processed.

RCODE — One-character field that contains the return code.

0 = successful
1 = unsuccessful (display station was not attached to program)
2 = unsuccessful (display station was not a requestor)

AREA — Field, up to 256 characters, that is used with SUBR21 to transfer the LDA. This field cannot be a data structure.

USING THE PROMPT OCL STATEMENT

The PROMPT OCL statement can be used to show a display directly from OCL without loading a program. The input returned from this display can be input to a subsequent display in the program or can be input to the procedure in which the PROMPT OCL statement occurred.

The format of the PROMPT OCL statement is as follows:

// PROMPT MEMBER–screen format load member name,FORMAT-display screen format name

$$\left[\text{,UPSI}-\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}\right]\ \left[\text{,PDATA}-\left\{\begin{matrix}\underline{\text{NO}}\\ \text{YES}\end{matrix}\right\}\right]$$

The PROMPT statement provides a return code to indicate which command or function key was pressed. This code can be checked by using the ?CD? substitution expression. The following chart shows the code returned for each key pressed:

Key	Return Code
Enter/Rec Adv	0000
Command key 01-24	2001-2024
Roll Up	2090
Roll Down	2091
Help	2092
Record Backspace	2093

This example shows the use of the PROMPT OCL statement and the return codes to set switches allowing the selective listing of library members when a particular command key is pressed by the operator. The use of the return codes to set switches, which indicate which library members are to be listed, is necessitated by the fact the SSP resets the return code to 0000 whenever it executes a RUN OCL statement.

```
LIBRARY DIRECTORY LISTING
   CURRENT SESSION LIBRARY    DRFLIB
   PROCEDURE MEMBERS ONLY      COMMAND KEY 1
   SOURCE MEMBERS ONLY         COMMAND KEY 2
   OBJECT MEMBERS ONLY         COMMAND KEY 3
   SUBROUTINE MEMBERS ONLY     COMMAND KEY 4
   SYSTEM DIRECTORY ONLY       COMMAND KEY 5
   ALL DIRECTORIES             COMMAND KEY 6

        CANCEL REQUEST         COMMAND KEY 7
```

```
// SWITCH 00000000     ?1F'?SLIB?'?
// TAG PROMPT
// PROMPT MEMBER-EXAMPLE,FORMAT-EXAMPLE,UPSI-YES
// IFF ?CD?>0000  GOTO PROMPT
// IF ?CD?/2007 CANCEL
// IF ?CD?/2001 SWITCH 10000000
// IF ?CD?/2002 SWITCH 01000000
// IF ?CD?/2003 SWITCH 00100000
// IF ?CD?/2004 SWITCH 00010000
// IF ?CD?/2005 SWITCH 00001000
// IF ?CD?/2006 SWITCH 00000100
// LOAD $MAINT
// RUN
// COPY FROM-?1?,
// IF SWITCH1-1 LIBRARY-P,
// IF SWITCH2-1 LIBRARY-S,
// IF SWITCH3-1 LIBRARY-O,
// IF SWITCH4-1 LIBRARY-R,
// IF SWITCH5-1 LIBRARY-SYSTEM,
// IF SWITCH6-1 LIBRARY-ALL,
//    NAME-DIR,TO-PRINT
// END
```

Initialize switches and parameter 1, which contains the session library.

Command key 1 was pressed; set switch 1.

Command key 2 was pressed; set switch 2.

Command key 3 was pressed; set switch 3.

Command key 4 was pressed; set switch 4.

Command key 5 was pressed; set switch 5.

Command key 6 was pressed; set switch 6.

Select correct parameter based upon which switch was set.

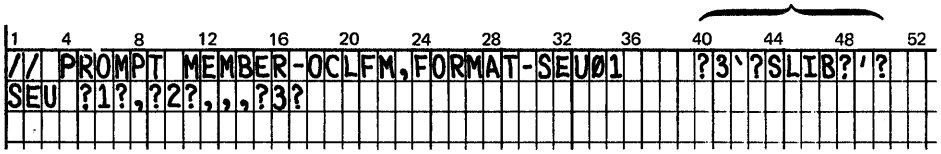
Note: The PROMPT-OCL statement cannot be used with a format that requires more than 88 characters of execution output data. If PDATA-NO or the PDATA parameter is not specified, the PROMPT OCL statement cannot be used to display a screen that has more than 88 characters of input data. If PDATA-YES is specified, the maximum amount of input data is controlled by the user program that reads the screen.

Using the PROMPT OCL Statement with PDATA-NO

PDATA-NO is the default value for the PDATA parameter and specifies that all input from the display is used for parameters in the procedure. The input data is inserted into positional parameters 1 through 11 in sequence. Each parameter contains eight bytes; therefore, the input/output data on the prompt screen can contain up to 88 characters. Parameters can be used for subsequent OCL processing or passed as data to other procedures or programs.

As an example of using the PROMPT statement with PDATA-NO, consider a user-written procedure called SEUP, which prompts the operator for the member name, member type, and library of a member to be edited with SEU. The PROMPT screen displayed by SEUP will show the active session library as the default library to be used. The statements in SEUP are:

Defaults the session library name to parameter 3.



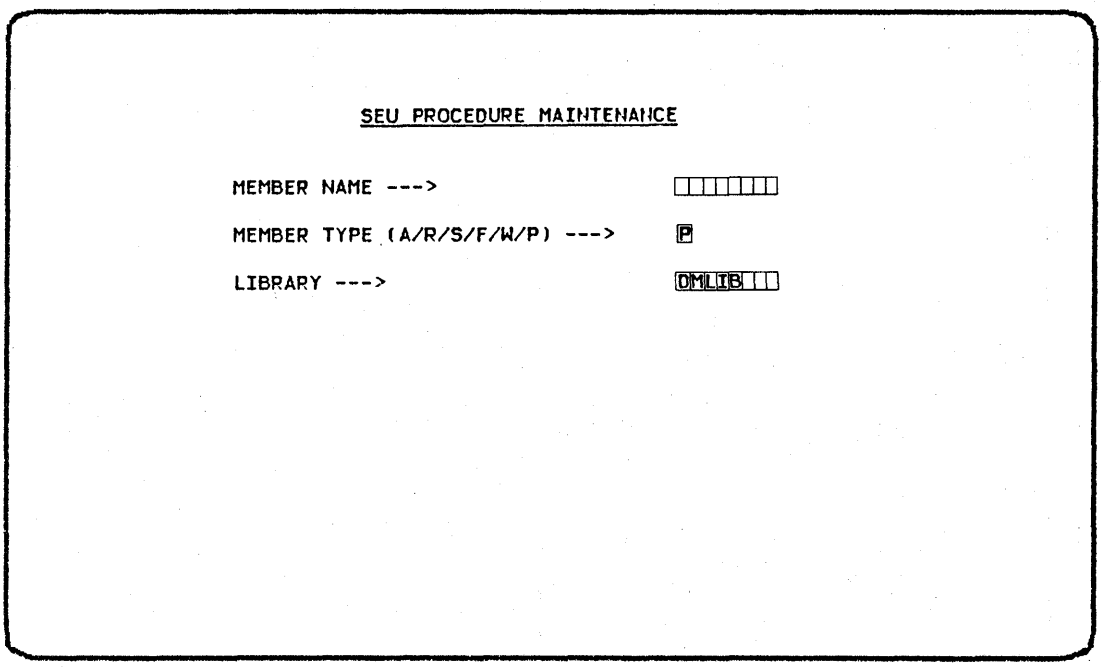
Notice that parameter 3, if it is not coded on the procedure command, is assigned the value of the current session library. Because all substitutions in a record are performed before the record is sent to the initiator function for processing, the assignment of the session library to parameter 3 is performed before the prompt screen is displayed.

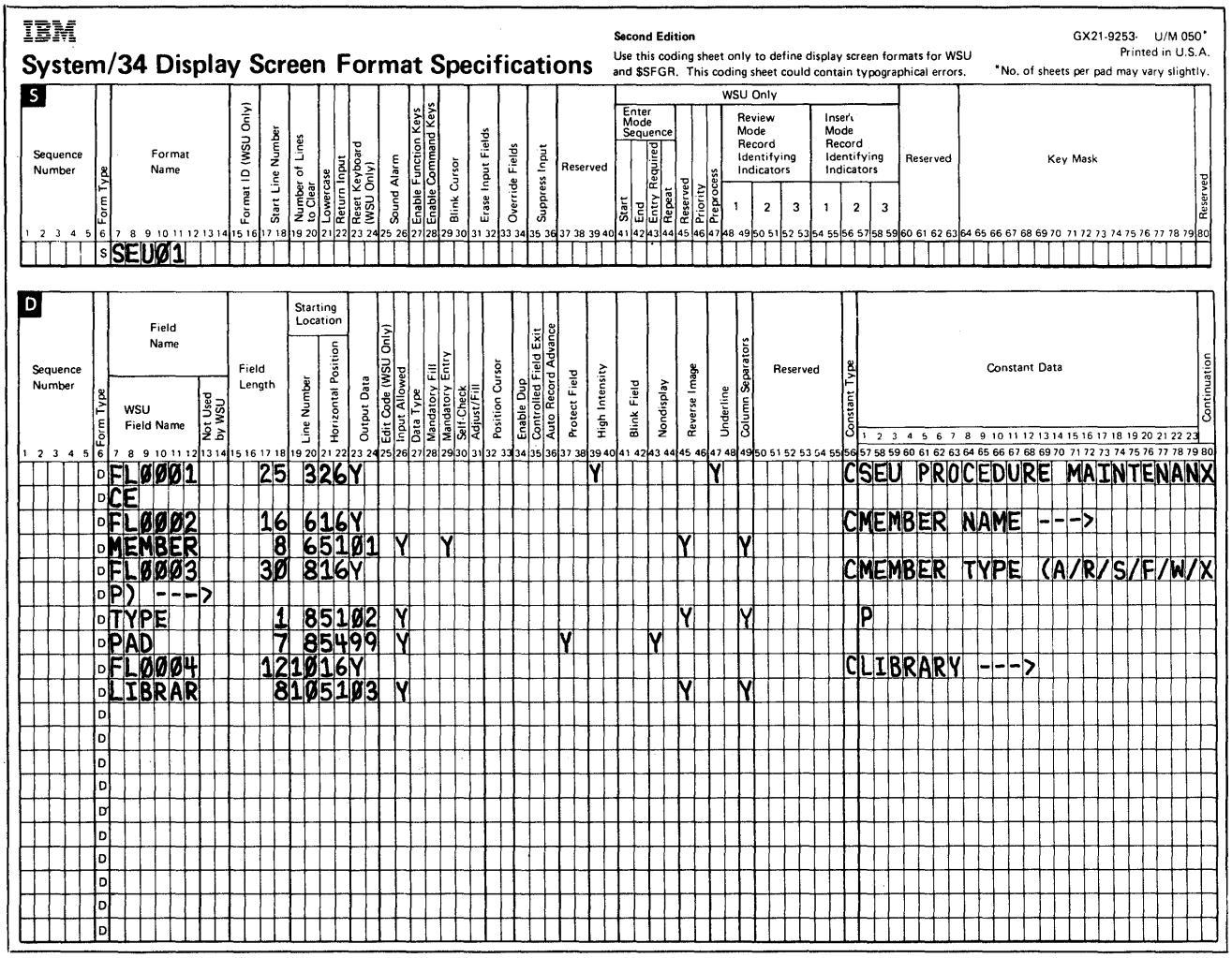
If a parameter is specified before the prompt screen is displayed, the corresponding SFGR indicator is set on. For example, if parameter 2 is specified, SFGR indicator 02 is set on.

Using the UPSI Parameter of the PROMPT OCL Statement

When UPSI-YES is specified on the PROMPT OCL statement, the settings of the UPSI switches affects the SFGR indicators. Each of the UPSI switches U1 through U8 that is on sets on the corresponding SFGR indicator 91 through 98. This allows control of the display by the setting of the UPSI switches in a previous program or SWITCH OCL statement.

The format for the prompt screen is called SEU01 and is in a format load member called OCLFM. The prompt screen and its S and D specifications are as follows:





IBM

System/34 Display Screen Format Specifications

Second Edition

Use this coding sheet only to define display screen formats for WSU and $SFGR. This coding sheet could contain typographical errors.

GX21-9253· U/M 050*
Printed in U.S.A.
*No. of sheets per pad may vary slightly.

S

Sequence Number	Form Type	Format Name	Format ID (WSU Only)	Start Line Number	Number of Lines to Clear	Lowercase	Return Input	Reset Keyboard (WSU Only)	Sound Alarm	Enable Function Keys	Enable Command Keys	Blink Cursor	Erase Input Fields	Override Fields	Suppress Input	Reserved	WSU Only: Enter Mode Sequence: Start	End	Entry Required	Repeat	Reserved	Priority	Preprocess	WSU Only: Review Mode Record Identifying Indicators: 1	2	3	WSU Only: Insert Mode Record Identifying Indicators: 1	2	3	Reserved	Key Mask	Reserved
1 2 3 4 5	6	7 8 9 10 11 12 13 14	15 16	17 18	19 20	21	22	23 24	25 26	27	28	29 30	31 32	33 34	35 36	37 38 39 40	41	42	43	44	45	46	47	48 49	50 51	52 53	54 55	56 57	58 59	60 61 62 63	64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	S	SEU01																														

D

Sequence Number	Form Type	Field Name: WSU Field Name	Field Name: Not Used by WSU	Field Length	Starting Location: Line Number	Starting Location: Horizontal Position	Output Data	Edit Code (WSU Only)	Input Allowed	Data Type	Mandatory Fill	Mandatory Entry	Self Check	Adjust/Fill	Position Cursor	Enable Dup	Controlled Field Exit	Auto Record Advance	Protect Field	High Intensity	Blink Field	Nondisplay	Reverse Image	Underline	Column Separators	Reserved	Constant Type	Constant Data	Continuation
1 2 3 4 5	6	7 8 9 10 11 12	13 14	15 16 17 18	19 20	21 22	23 24	25	26	27	28	29	30	31	32	33	34	35	36	37 38	39 40	41 42	43 44	45 46	47 48	49	50 51 52 53 54 55	56	57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	D	FL0001		25	326		Y													Y				Y			C	SEU PROCEDURE MAINTENANX	
	D	CE																											
	D	FL0002		16	616		Y																				C	MEMBER NAME --->	
	D	MEMBER		8	651	01	Y		Y													Y		Y					
	D	FL0003		30	816		Y																				C	MEMBER TYPE (A/R/S/F/W/X	
	D	P) --->																											
	D	TYPE		1	851	02	Y													Y				Y			P		
	D	PAD		7	854	99	Y												Y			Y							
	D	FL0004		12	1016		Y																				C	LIBRARY --->	
	D	LIBRAR		8	1051	03	Y													Y				Y					
	D																												
	D																												
	D																												
	D																												
	D																												
	D																												
	D																												
	D																												
	D																												

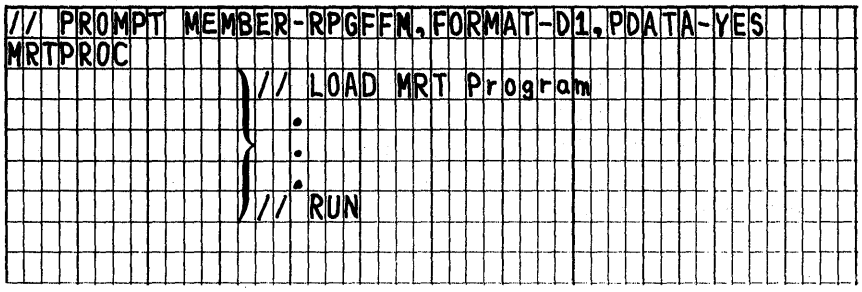
Each parameter requires eight bytes of input data from the screen. In this case, however, the operator needs to enter only one character for the second parameter (member type); therefore, a 7-byte pad field is defined. A dummy indicator is coded in columns 23 and 24 of the D specification for the pad field. The dummy indicator can be any value greater than 11. In this case, the value is 99. The pad field is protected and not displayed. This field is skipped and the operator cannot enter characters into this field. The value assigned to the second parameter in the procedure will be the value entered by the operator followed by the seven blanks from the pad field.

Using the PROMPT OCL Statement with PDATA-YES

The PDATA parameter, if yes, specifies that input from the display is read by a user program on the first input request. OCL processing does not stop as it does when PDATA-NO is specified.

Program initiation time can cause a few seconds' delay before the operator sees the first display in the program. Using the PROMPT OCL statement with the PDATA-YES parameter to show the first display can minimize this delay because an operator can enter data on the display while the program is initiated. When it is initiated, the program waits for the operator to finish his data entry and then reads from the display.

The PROMPT OCL statement can also be used to show the initial display for an MRT program. The PROMPT statement must be used in an SRT procedure that calls an MRT procedure. For example:



In the previous example, PROCA shows display D1 and then calls MRTPROC, which is an MRT procedure. Input from display D1 can then be read by the MRT program.

The sample application described in Chapter 5 also uses the PROMPT OCL statement to show the initial display.

PROTECTING RECORDS FROM CONCURRENT UPDATES IN AN MRT PROGRAM

System/34 protects files that are shared by two or more programs by not allowing concurrent reads for updates to the same sector of data. The system cannot, however, protect a single program from itself or two MRT programs from each other; this means that a program that allows concurrent updates to a file by two or more display stations might produce incorrect results.

Situations for which a program might need this internal protection arise most often in RPG MRT programs. WSU has built-in protection of its transaction file so that an operator can see and change only those entries that were made from his display station. RPG MRT programs do not have this built-in protection.

The following information describes a coding technique for protecting records within an RPG MRT program or COBOL MRT program. This technique protects an indexed file and, therefore, record keys are referenced. You can use the same method to protect direct files. In that case, the references to record keys would apply to relative record numbers.

This coding technique uses two subroutines, RESERV and RELEAS, and a table, TABREC. The table is used to hold the key of each protected record. The table elements, therefore, must be the same length as the key. Because a display station can own only one record at a time, the number of elements in the table should be the same as the maximum number of requestors of the program.

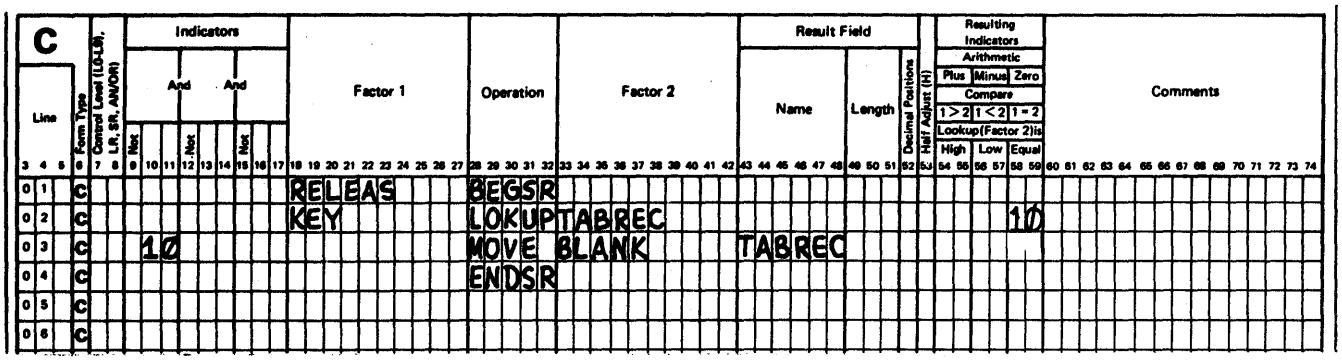
The following subroutine, RESERV, is executed each time a record is read for a possible update:

C	Line	Form Type	Control Level (L0-L9), LR, SR, AN/OR)	Indicators: And	Indicators: And	Indicators: And	Factor 1	Operation	Factor 2	Result Field: Name	Result Field: Length	Result Field: Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus	Resulting Indicators: Arithmetic: Minus	Resulting Indicators: Arithmetic: Zero	Resulting Indicators: Compare: 1>2	Resulting Indicators: Compare: 1<2	Resulting Indicators: Compare: 1=2	Resulting Indicators: Lookup(Factor 2)is: High	Resulting Indicators: Lookup(Factor 2)is: Low	Resulting Indicators: Lookup(Factor 2)is: Equal	Comments
	3 4 5	6	7 8	Not 9	10 11	Not 12 13 14	Not 15 16 17	18 19 20 21 22 23 24 25 26 27	28 29 30 31 32	33 34 35 36 37 38 39 40 41 42	43 44 45 46 47 48	49 50 51	52	53	54 55	56 57	58 59	60 61 62 63 64 65 66 67 68 69 70 71 72 73 74						
01		C						RESERV	BEGSR														
02		C						KEY	LOKUP	TABREC							11						
03		C		N11				BLANK	LOKUP	TABREC							10						
04		C		10					MOVE	KEY		TABREC											
05		C							ENDSR														
06		C																					
07		C																					

The field named BLANK is defined in the program as a blank field that has the same length as the key. You can also use the figurative constant *BLANK, which always has the length of the receiving field.

The RESERV subroutine checks whether or not the requested record is in use. If this key is in the table, the record is in use, and indicator 11 turns on. The RPG program should ask the operator to retry his request and, thus, give the owner of the record a chance to release it. If the key is not in the table, the record is not in use. The subroutine finds a blank element in the table in which to store the key. Putting the key in the table establishes protection for the record.

The following subroutine, RELEAS, is executed when an operator is finished with a record. The subroutine finds the key in the table and then sets the element to blanks to remove protection from that record and make it available to other display stations.



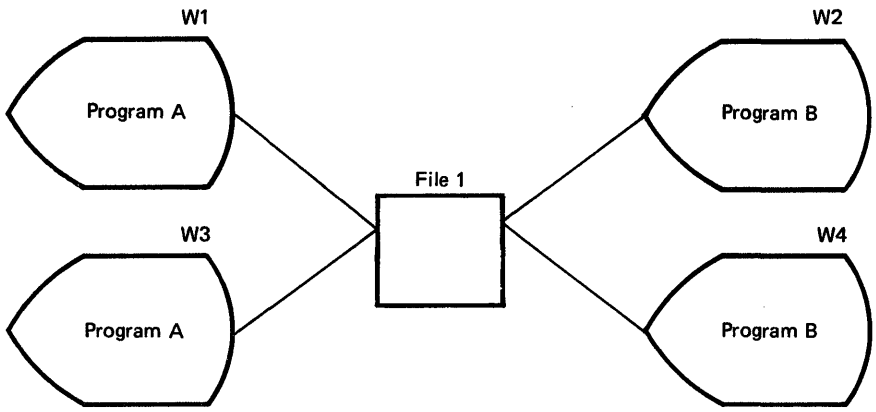
This technique of using the table and two subroutines protects one file. To protect multiple files used by one RPG MRT program, you must provide one table per file. If the lengths of the elements differ among tables, you also need a BLANK field for each table. You can also use the figurative constant *BLANK, which always has the length of the receiving field.

You should have record lengths that are a multiple of 256 bytes when using this technique. Refer to *File Concepts* in Chapter 2 for additional information about file sharing.

PROTECTING RECORDS FROM CONCURRENT UPDATES BY MULTIPLE MRT PROGRAMS

When two or more MRT programs share a file and are allowed to update it, unexpected results can occur if you do not protect records from concurrent updates.

For example, assume that two operators at displays W1 and W3 are using MRT Program A to update File 1. At the same time, two operators at displays W2 and W4 are using MRT Program B to update the same file.



To show how an unexpected result can occur in this situation, assume that while W1 reads record 14 and displays an on-hand quantity, W3 reads record 60. System protection is removed from the sector that contains the first record read (record 14) and is given to the sector that contains the last record read (record 60). Because of this loss of protection, the on-hand quantity in record 14 could be read and updated by another program. Program logic must be able to handle this situation. For example, when a field is read and displayed, its value on disk rather than its value on the display should be used for subsequent calculations, because the disk value is more current.

When two programs allocate inventory based on the same displayed on-hand quantity, one of the operators may make an incorrect decision because the quantity he sees is not most current. An editing routine in the program should display a message when an on-hand quantity is not sufficient.

If two MRT programs share a file and both can update that file, you can protect records from concurrent updates by adding and using an ownership field in each record. If possible, this field should be large enough to hold the ID of the display station that updates the record.

The ownership field should be blank until it becomes owned (read for an update) by a program for a particular display station. To establish ownership, a program should place its name and a display station ID in the ownership field. If that field is already owned by another display station, a message could be displayed indicating that the record is temporarily not available. The operator could decide to reread the record or continue with other calculations and return later to reread the record.

A program could remove ownership of a record by setting the ownership field to blank; this usually would be done when the updated record is written.

This technique simplifies file recovery because the ownership field provides a good picture of what was happening when an error occurred. If this technique is used, a recovery program is needed to check and reset the ownership field in all records whenever recovery is necessary, such as after a power failure.

An ownership field can also be used to prevent an operator from incorrectly updating the same record using two different programs. For example, assume that the operator requests a program, begins updating a record, and then cancels the program using the Attn key and option 2 or 3 on the Inquiry display. If the operator requests the same record using another program, the ownership field would indicate that the canceled program still owned the record. The current program could allow an option to be selected that overrides the program name of the ownership field or could display a message instructing the operator to recall the first program and normally complete the transaction update.

USING THE LOCAL DATA AREA TO INCREASE SORT PROGRAM FLEXIBILITY

The System/34 local data area (LDA) provides a way to increase the flexibility of System/34 sort programs. By initializing the LDA through OCL and then accessing this area in the sort statements, you can allow one sort source member to serve several functions.

For example, suppose the item master file should be sorted to include:

- All items
- Only certain items
- Items within a range

The procedure format might be:

SORTITEM { ALL
ITEMS, Item-1, item-2,...item-9
RANGE, low item, high item }

The following procedure could be coded for this sort:

```
1   4       8       12      16      20      24      28      32      36      40      44      48      52
// IF ?1?/ALL LOCAL OFFSET-1,DATA-'I*CI*CI*C                                                    '
// IF ?1?/RANGE LOCAL OFFSET-1,DATA-'I CIACI*CGELE'
// IF ?1?/ITEMS LOCAL OFFSET-1,DATA-'I CIOCIOCEQEQ'
// LOAD #GSORT
// FILE NAME-INPUT,LABEL-ITEMMST
// FILE NAME-OUTPUT,LABEL-ITMADR,RECORDS-1000
// RUN
      HSORTA      6A
      ?L'1,3'?    6  11?L'10,2'?C?2?
      ?L'4,3'?    6  11?L'12,2'?C?3?
      ?L'7,3'?    6  11?L'12,2'?C?4?
      ?L'7,3'?    6  11?L'12,2'?C?5?
      ?L'7,3'?    6  11?L'12,2'?C?6?
      ?L'7,3'?    6  11?L'12,2'?C?7?
      ?L'7,3'?    6  11?L'12,2'?C?8?
      ?L'7,3'?    6  11?L'12,2'?C?9?
      ?L'7,3'?    6  11?L'12,2'?C?10?
      FNC   6  11
// END
```

As the previous procedure shows, you can control calls of the sort program via parameters; such control is particularly useful when you select fields based on variable data in a field.

As the following example shows, you can substitute nearly all sort specifications:

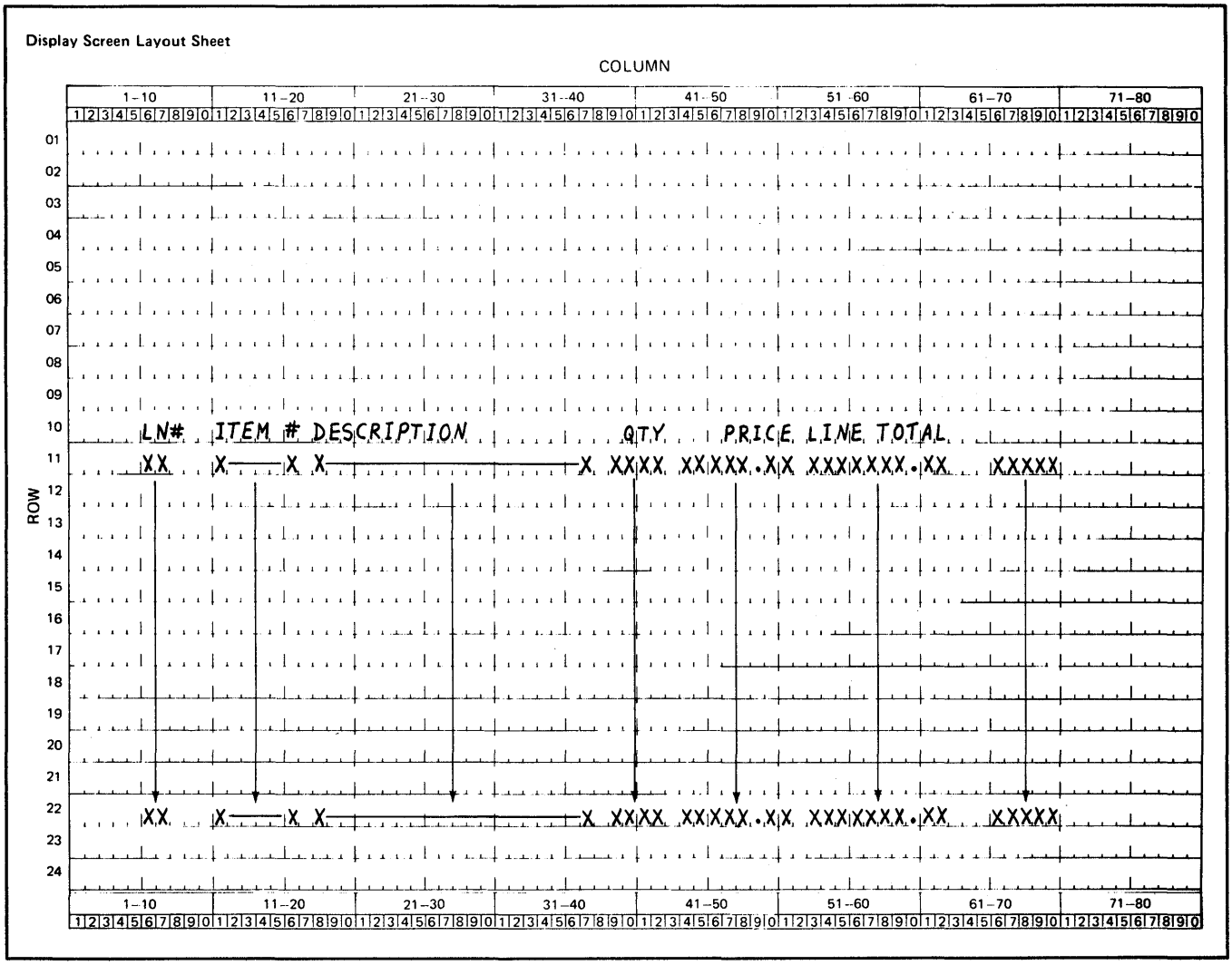
```
1   4   8   12  16  20  24  28  32  36  40  44  48  52
//  IF ?1?/ITEM LOCAL OFFSET-1,DATA-'0000600060011'
//  IF ?1?/VENDOR LOCAL OFFSET-1,DATA-'0000200040005'
//  LOAD #GSORT
//  FILE NAME-INPUT,LABEL-ITEMMST
//  FILE NAME-OUTPUT,LABEL-ITMADR
//  RUN
        HSORTA ?L'1,5'?A
        FNC?L'6,8'?
//  END
```

By putting the length of the control field and its beginning and ending positions in the record in the LDA, you can pass this data to the procedure that calls the sort program.

Note that the ?L'x,y'? option truncates all leading and trailing blanks. Therefore, any sort specification fields used in this manner should be zero-filled (as are the item numbers in the first example).

USING DATA STRUCTURES FOR MULTIPLE LINE DISPLAYS

Using data structures in RPG programs can reduce the number of lines you code and the amount of array processing required to display multiple lines that have the same format. For example, a data structure might be used in an RPG program that has the following display:



IBM

System/34 Display Screen Format Specifications

Second Edition

Use this coding sheet only to define display screen formats for WSU and $SFGR. This coding sheet could contain typographical errors.

GX21-9253 U/M 050*
Printed in U.S.A.
*No. of sheets per pad may vary slightly.

S

Sequence Number	Form Type	Format Name	Format ID (WSU Only)	Start Line Number	Number of Lines to Clear	Lowercase	Return Input	Reset Keyboard (WSU Only)	Sound Alarm	Enable Function Keys	Enable Command Keys	Blink Cursor	Erase Input Fields	Override Fields	Suppress Input	Reserved	WSU Only: Enter Mode Sequence: Start	End	Entry Required	Repeat	Reserved	Priority	Preprocess	Review Mode Record Identifying Indicators: 1	2	3	Insert Mode Record Identifying Indicators: 1	2	3	Reserved	Key Mask	Reserved
1 2 3 4 5	6	7 8 9 10 11 12 13 14	15 16	17 18	19 20	21	22	23 24	25 26	27	28	29 30	31 32	33 34	35 36	37 38 39 40	41	42	43	44	45	46	47	48 49	50 51	52 53	54 55	56 57	58 59	60 61 62 63	64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	S	INVOICE		01	22																										

D

Sequence Number	Form Type	Field Name: WSU Field Name	Not Used by WSU	Field Length	Starting Location: Line Number	Horizontal Position	Output Data	Edit Code (WSU Only)	Input Allowed	Data Type	Mandatory Fill	Mandatory Entry	Self-Check	Adjust/Fill	Position Cursor	Enable Dup	Controlled Field Exit	Auto Record Advance	Protect Field	High Intensity	Blink Field	Nondisplay	Reverse Image	Underline	Column Separators	Reserved	Constant Type	Constant Data	Continuation
1 2 3 4 5	6	7 8 9 10 11 12	13 14	15 16 17 18	19 20	21 22	23 24	25	26	27	28	29	30	31 32	33	34	35	36	37 38	39 40	41 42	43 44	45 46	47 48	49	50 51 52 53 54 55	56	57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	D			23	10	6	Y																				LNO ITEM # DESCRIPTION	
	D			23	10	40	Y																				QTY PRICE LINE TOTAL	
	D	OLN1		65	11	6	Y																					
	D	OLN2		65	12	6	Y																					
	D	OLN3		65	13	6	Y																					
	D	OLN4		65	14	6	Y																					
	D	OLN5		65	15	6	Y																					
	D	OLN6		65	16	6	Y																					
	D	OLN7		65	17	6	Y																					
	D	OLN8		65	18	6	Y																					
	D	OLN9		65	19	6	Y																					
	D	OLN10		65	20	6	Y																					
	D	OLN11		65	21	6	Y																					
	D	OLN12		65	22	6	Y																					
	D																											
	D																											
	D																											
	D																											
	D																											
	D																											

This section provides a partial RPG program that uses a data structure to show as many as 12 lines of an open order on the previous display.

On each line of the display, the following fields are shown:

Field Name	Description	Length
LNNO	Line number	2
ITNBR	Item number	6
DESCR	Item description	20
QTYOR	Quantity on order	4
PRICE	Unit price	8
EXTEND	Extended price	10
THSRRN	This relative record number	5

The sample RPG program uses an array, OLN, for the fields on the display. The number of elements in the array is 12, which is the number of lines used on the display. The length of each element in the array is 65, which is the sum of the lengths of one line of fields and includes blanks between the fields.

The following sample RPG program uses a data structure that corresponds to the format of a line on the display:

RPG CONTROL AND FILE DESCRIPTION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9092 UM/050*
Printed in U.S.A.

Program		Keying Instruction	Graphic								Card Electro Number
Programmer	Date		Key								

Page [ ][ ] of [ ] Program Identification [ ][ ][ ][ ][ ][ ][ ][ ]

Control Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

H

Line	Form Type	Size to Compile	Object Output	Listing Options	Size to Execute	Debug	Reserved	Currency Symbol	Date Format	Date Edit	Inverted Print	Reserved	Number of Print Positions	Alternate Collating Sequence	Reserved	Inquiry	Reserved	Sign Handling	1P Forms Position	Indicator Setting	File Translation	Punch MFCU Zeros	Nonprint Characters	Reserved	Table Load Halt	Shared I/O	Field Print	Formatted Dump	RPG II Conversion	Number of Formats	S/3 Conversion	Subprogram	CICS/DL/I	Transparent Literal
3 4 5	6	7 8 9	10	11	12 13 14	15	16 17	18	19	20	21	22	23 24 25	26	27 28 29 30 31 32 33 34 35 36	37	38 39	40	41	42	43	44	45	46	47	48	49	50	51	52 53	54	55	56	57
0 1	H																																

File Description Specifications

For the valid entries for a system, refer to the RPG reference manual for that system.

F

Line	Form Type	Filename	I/O/U/C/D	P/S/C/R/T/D/F	E	A/D	F/V/S/M/D/E	Block Length	Record Length	L/R	A/P/I/K	1/X/D/T/R/ or 2	Type of File Organization or Additional Area	Overflow Indicator	Key Field Starting Location	Extension Code E/L	Device	Symbolic Device	Labels S/N/E/M	Name of Label Exit	Extent Exit for DAM	Storage Index	K	Option	Entry	A/U	Number of Tracks for Cylinder Overflow	Number of Extents	Tape Rewind	File Condition U1-U8, UC	R/U/N
3 4 5	6	7 8 9 10 11 12 13 14	15	16	17	18	19	20 21 22 23	24 25 26 27	28	29 30	31 32 33 34	35 36 37 38	39			40 41 42 43 44 45 46	47 48 49 50 51 52	53	54 55 56 57 58 59	60 61 62 63 64 65					66	67 68 69	70	71 72	73 74	
0 2	F	WORKSTN	C	P			F		1500								WORKSTN															
0 3	F	*																														
0 4	F	ORDDET	I	C			F		64								DISK															
0 5	F	*																														
0 6	F	ITEMMST	I	C			F		256	R	6	A	I		3		DISK															
0 7	F																															
0 8	F																															
0 9	F																															
1 0	F																															
	F																															
	F																															

72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

*Number of sheets per pad may vary slightly.

RPG EXTENSION AND LINE COUNTER SPECIFICATIONS

GX21-9091 UM:050*
Printed in U.S.A.

IBM International Business Machines Corporation

Program		Keying Instruction	Graphic								Card Electro Number
Programmer	Date		Key								

Page 1 2 of __ Program Identification 75 76 77 78 79 80

Extension Specifications

E

Line	Form Type	Record Sequence of the Chaining File	Number of the Chaining Field	From Filename	To Filename	Table or Array Name	Number of Entries Per Record	Number of Entries Per Table or Array	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Table or Array Name (Alternating Format)	Length of Entry	P/B/L/R	Decimal Positions	Sequence (A/D)	Comments
01	E			ARRAY ELEMENT LENGTH IS NO. OF DISPLAY POSITIONS ON ONE LINE														
02	E			NO. OF ELEMENTS IN ARRAY IS THE NUMBER OF DISPLAY LINES USED														
03	E																	
04	E					OLN		12	65									
05	E																	
06	E																	
07	E																	
08	E																	
	E																	
	E																	

Line Counter Specifications

L

Line	Form Type	Filename	1 Line Number	1 FL or Channel Number	2 Line Number	2 OL or Channel Number	3 Line Number	3 Channel Number	4 Line Number	4 Channel Number	5 Line Number	5 Channel Number	6 Line Number	6 Channel Number	7 Line Number	7 Channel Number	8 Line Number	8 Channel Number	9 Line Number	9 Channel Number	10 Line Number	10 Channel Number	11 Line Number	11 Channel Number	12 Line Number	12 Channel Number
11	L																									
12	L																									
	L																									

*Number of sheets per pad may vary slightly.

I

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name	Record Identification Codes 1 Position	1 Not (N)	1 C/Z/D	1 Character	2 Position	2 Not (N)	2 C/Z/D	2 Character	3 Position	3 Not (N)	3 C/Z/D	3 Character	Stacker Select	P/B/L/R	Field Location From	Field Location To	Data Structure Occurs n Times	Data Structure Length	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators Plus	Field Indicators Minus	Field Indicators Zero or Blank
		Data Structure Name	O A / R N / D																														
01	I	* ORDER DETAIL FILE - LINE ITEM RECORD																															
02	I	ORDDET NS				Ø3		1	CL		2	CI																					
03	I																						3		80	ORDNO							
04	I																						9		14	CUSTNO							
05	I																						15		18	QTYOR							
06	I																						19		24	ITNBR							
07	I																						25		31	PRICE							
08	I																						25		29	UDOLAR							
09	I																						30		31	UCENTS							
10	I																						32		33	LNNO							
11	I																						61		64	THSRN							
12	I	* ITEM MASTER FILE - ITEM RECORD																															
13	I	ITEMMST NS				Ø5		1	CM		2	CB																					
14	I																						9		28	DESCR							
15	I	WORKSTN NS				Ø1																											
16	I																																

IBM International Business Machines Corporation

RPG INPUT SPECIFICATIONS

GX21-9094 UM 050*
Printed in U.S.A.

Program		Keying Instruction	Graphic		Card Electro Number
Programmer	Date		Key		

Page 1 2 of ___ Program Identification 75 76 77 78 79 80

Line	Form Type	Filename or Record Name	Sequence	Number (1/N), E	Option (O), U, S	Record Identifying Indicator, **, or DS	External Field Name / Record Identification Codes 1: Position	1: Not (N)	1: C/Z/D	1: Character	2: Position	2: Not (N)	2: C/Z/D	2: Character	3: Position	3: Not (N)	3: C/Z/D	3: Character	Stacker Select	P/B/L/R	Field Location: From	Field Location: To	Data Structure: Occurs n Times	Data Structure: Length	Decimal Positions	RPG Field Name	Control Level (L1-L9)	Matching Fields or Chaining Fields	Field Record Relation	Field Indicators: Plus	Field Indicators: Minus	Field Indicators: Zero or Blank
		Data Structure Name	O A / R N / D																													
01	I	* ORGANIZE				DATA	STRUCTURE				TO				REFLECT						ONE			DISPLAY		LINE							
02	I					DS																											
03	I																				1			20	LNNO							
04	I																				6			110	ITNBR							
05	I																				13			32	DESCR							
06	I																				34			370	QTYOR							
07	I																				39			430	UDOLAR							
08	I																				44			44	DOT1							
09	I																				45			462	UCENTS							
10	I																				48			54	EDOLAR							
11	I																				55			55	DOT2							
12	I																				56			57	ECENTS							
13	I																				61			650	THSRRN							
14	I																				1			65	OUTLIN							
15	I																															
16	I																															
17	I																															
18	I																															
19	I																															
20	I																															
	I																															
	I																															
	I																															
	I																															
	I																															

72 71 70 69 68 67 66 65 64 63 62 61 60 59 58 57 56 55 54 53 52 51 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

*Number of sheets per pad may vary slightly.

RPG CALCULATION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9093 UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic Key | Card Electro Number

Page 01 of 02 | Program Identification

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
010C	*		INITIALIZE EDIT CHARACTERS									
020C			N99		MOVE	'.'	DOT1					
030C			N99		MOVE	'.'	DOT2					
040C			N99		SETON						99	
050C												
060C												
070C												
080C	*		INITIALIZE ARRAY INDEX									
090C					Z-ADD	1	X		20			
100C					Z-ADD	1	NXTRRN		50			
110C				RTRN1	TAG							
120C	*		READ THE ORDER DETAIL FILE									
130C				NXTRRN	CHAIN	ORDDET					50	
140C			50		SETON						LR	
150C			50		GOTO	END						
160C	*		READ THE ITEM MASTER FILE FOR DESCRIPTION									
170C				ITNBR	CHAIN	ITEMMST					51	
180C			51		MOVE	'NO DESCR'	DESCR					
190C	*		CALCULATE EXTENDED PRICE & SET FOR OUTPUT									
200C				QTYOR	MULT	PRICE	EXTEND		92			
210C					MOVEL	EXTEND	EDOLAR		7			
230C					MOVE	EXTEND	ECENTS		2			
240C	*		MOVE DATA STRUCTURE NAME TO OUTPUT ARRAY ELEMENT									
250C					MOVE	OUTLIN	OLN,X					
260C												

*Number of sheets per pad may vary slightly.

RPG CALCULATION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9093 UM/050*
Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic Key | Card Electro Number

Page 02 of 02 | Program Identification

C

Line	Form Type	Control Level (L0-L9, LR, SR, AN/OR)	Indicators: And / And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic: Plus / Minus / Zero; Compare: 1>2 / 1<2 / 1=2; Lookup(Factor 2)is: High / Low / Equal	Comments
010C	*		CHECK TO SEE IF AT END IF 50 SETON				12					
020C				X	COMP	12					12	ARRAY IS FULL
030C			N12	X	ADD	1	X					NEXT ELEMENT
040C			N12	NXTRRN	ADD	1	NXTRRN					NEXT REL REC
050C			N12		GOTO	RTRN1						FOR NEXT LINE
060C					TAG	END						END OF PROGRAM
070C												
080C												
090C												

IBM International Business Machines Corporation

RPG OUTPUT SPECIFICATIONS

GX21 9090 UM/050*
Printed in U.S.A.

Program		Keying Instruction	Graphic								Card Electro Number
Programmer	Date		Key								

Page [1 2] of

Program Identification 75 76 77 78 79 80

O

Line	Form Type	Filename or Record Name	Type (H/D/T/E)	Space Before	Space After	Skip Before	Skip After	Output Indicators And	Output Indicators And	Field Name or EXCPT Name	Edit Codes	End Position in Output Record	P B L R	Constant or Edit Word
3 4 5	6	7 8 9 10 11 12 13 14 15	16	17	18	19 20	21 22	23 24 25	26 27 28 29 30 31	32 33 34 35 36 37 38	39	40 41 42 43	44	45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
0 1	O	WORKSTN						12						
0 2	O											K7		'INVOICE'
0 3	O									OLN	B	780		
0 4	O													
0 5	O													
0 6	O													
0 7	O													
0 8	O													
0 9	O													
1 0	O													
1 1	O													
1 2	O													
1 3	O													
1 4	O													
1 5	O													
1 6	O													
1 7	O													
1 8	O													
1 9	O													
2 0	O													
	O													
	O													
	O													
	O													
	O													

Commas	Zero Balances to Print	No Sign	CR		
Yes	Yes	1	A	J	X Remove Plus Sign
Yes	No	2	B	K	Y Date Field Edit
No	Yes	3	C	L	Z Zero Suppress
No	No	4	D	M	5 - 9 User Defined

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72

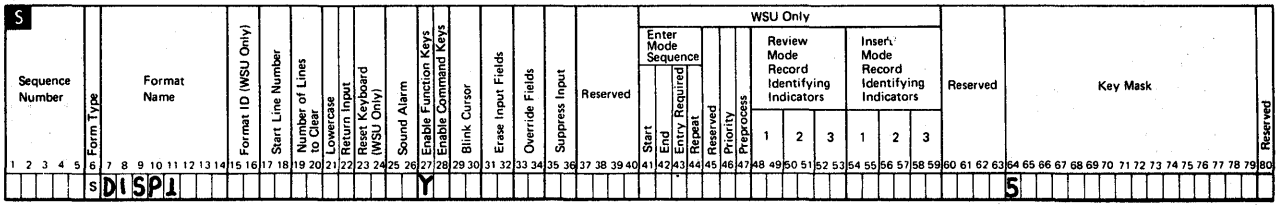
*Number of sheets per pad may vary slightly.

ACCESSING A FUNCTION CONTROL KEY OR COMMAND KEY IN AN RPG II PROGRAM

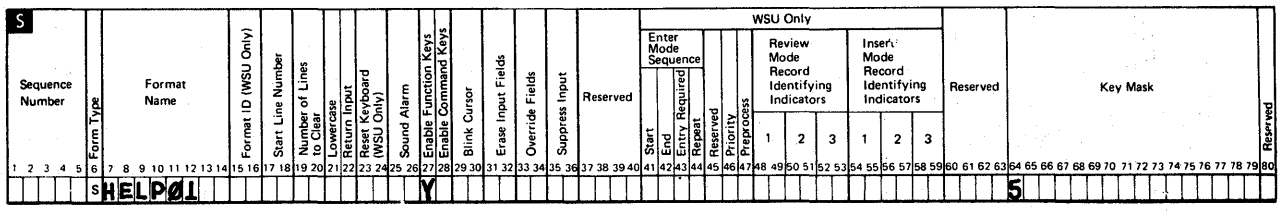
This section presents an example that briefly outlines the steps required to access a function control key or a command key within an RPG II program. In this case, the Help function control key is enabled and used in the program.

In the example program, a display called DISP1 is displayed. When DISP1 is displayed, the operator has the option of pressing the Help key for additional information. When the Help key is pressed, the program displays help screen called HELP01.

In the display screen format specifications for DISP1, a Y must be entered in column 27 and the key mask must be entered in columns 64 through 69 to enable the desired key(s). (In this example, a key mask of 5 indicates that the Help key is the only key enabled.) For further information about these specifications and other key mask values, refer to the description of the $SFGR utility program in Chapter 4 of the *SSP Reference Manual*. The following is the S specification for DISP1:

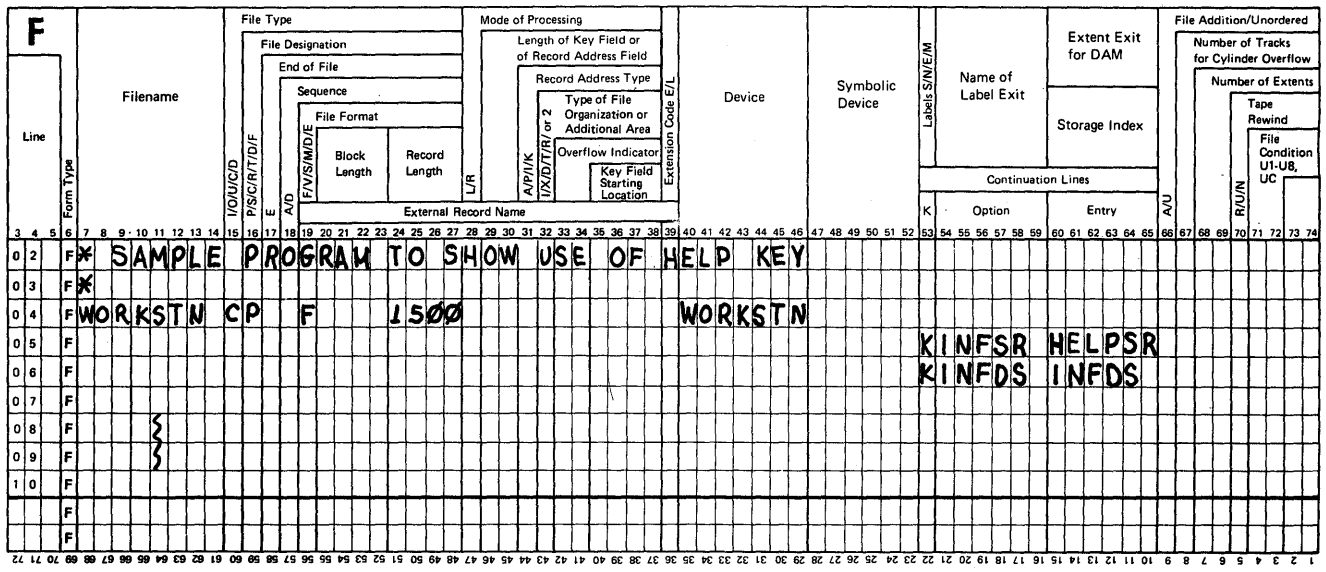


In the display screen format specifications for the help screen, HELP01, and for all other displays issued by the program, Y should also be specified in column 27 and a key mask specified. The following is the S specification for HELP01:



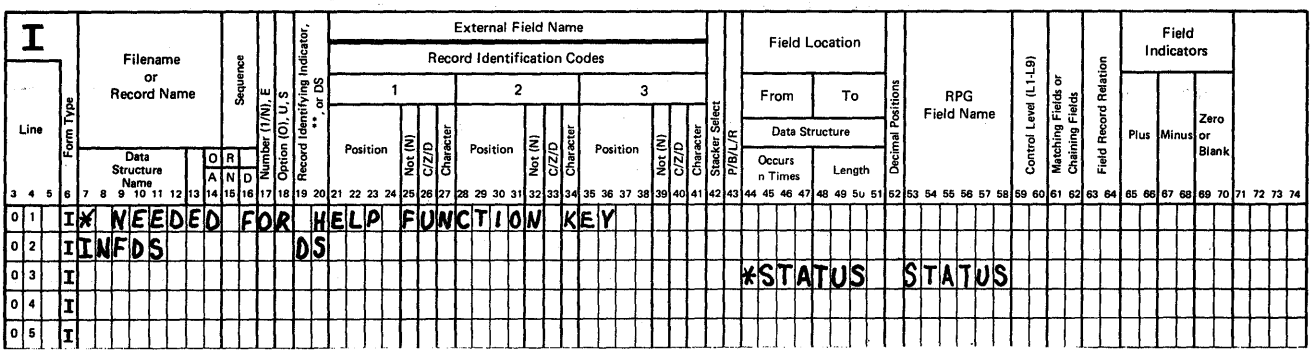
Within the RPG II program itself, two WORKSTN file continuation lines must be coded: One line identifies the INFDS data structure, and the other line identifies the INFSR subroutine that will receive control when the Help key is pressed. For complete information about the INFDS data structure and the INFSR subroutine, refer to Chapter 13 of the *RPG II Reference Manual*. The following are the WORKSTN F specifications for this example:

File Description Specifications



In this example, the INFSR subroutine is called HELPSR and the INFDS data structure is called INFDS.

The INFDS data structure is defined in the I specifications for the program:



The *STATUS subfield (shown as STATUS in this example) contains a 5-digit code that identifies the exception condition. In this case, the code 01125 indicates that the Help key was pressed.

The HELPSR subroutine, which gets control when the Help key is pressed, is coded in the C specifications for the program:

RPG CALCULATION SPECIFICATIONS

IBM International Business Machines Corporation

GX21-9093- UM/050* Printed in U.S.A.

Program | Programmer | Date | Keying Instruction | Graphic Key | Card Electro Number | Page 1 2 of | Program Identification 75 76 77 78 79 80

C

Line	Form Type	Control Level (L0-L9), LR, SR, AN/OR	Indicators: And / And	Factor 1	Operation	Factor 2	Result Field: Name	Length	Decimal Positions	Half Adjust (H)	Resulting Indicators: Arithmetic (Plus / Minus / Zero); Compare (1>2 / 1<2 / 1=2); Lookup (Factor 2) (High / Low / Equal)	Comments
01	C	*	ERROR SUBROUTINE:	CONTROL	PASSED HERE WHEN						HELP'	15	PRESSED
02	C			HELPSR	BEGSR								
03	C	*											
04	C			STATUS	COMP	01125						99	HELP KEY
05	C		N99		GOTO	NOTHLP							
06	C	*	OTHER FUNCTION KEYS	COULD BE	CHECKED AS	FOLLOWS:							
07	C	*		STATUS	COMP	01121						21	PRINT KEY
08	C	*		STATUS	COMP	01122						22	ROLL UP
09	C	*		STATUS	COMP	01123						23	ROLL DOWN
10	C	*		STATUS	COMP	01124						24	CLEAR
11	C	*		STATUS	COMP	01126						26	REC BACKSPACE
12	C				EXCPT								
13	C				SETOF						99		
14	C			NOTHLP	ENDSR	*GETIN'							BEGIN NEW CYCLE
15	C												
16	C												
17	C												
18	C												
19	C												
20	C												
	C												
	C												
	C												
	C												
	C												

*Number of sheets per pad may vary slightly.

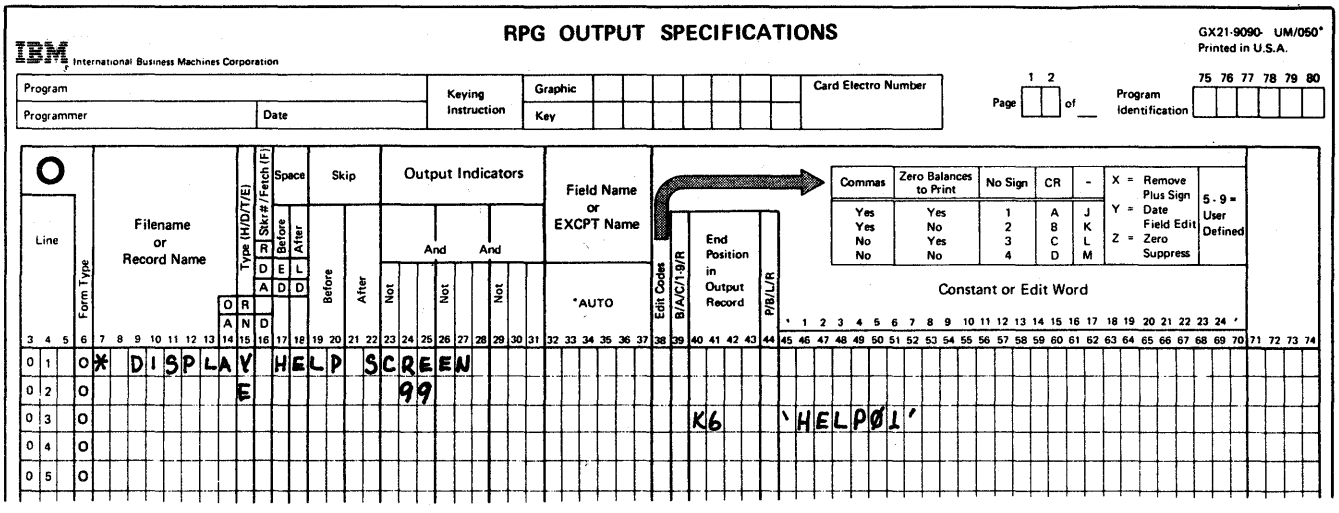
In this example, the subroutine compares the value in STATUS to 01125 to determine if the Help key was pressed. If the Help key was pressed, indicator 99 is on. When the EXCPT operation is processed, the HELP01 screen is displayed. The operator can then read and use the information on the display. Assuming there is only one help screen, the operator then presses the Enter/Rec Adv key, and the rest of the calculations are processed.

Note: If the operator enters data on a display and then presses the Help key, the entered data is lost when the screen is redisplayed. The screen is redisplayed in its original form.

At the end of this subroutine, factor 2 of the ENDSR operation causes control to return to the beginning of a new cycle. Possible values for factor 2 are:

Factor 2	Description
blank	Control passes to the RPG II error handling routine, which in most cases causes the program to halt.
'*GETIN'	Control returns to the beginning of a new cycle.
'*DETC'	Control returns to the beginning of detail calculations.
'*CANCL'	Files are closed and the program is canceled.

The following are the O specifications for displaying the HELP01 screen:



Chapter 5. Sample Applications

Sample Order Entry Application

This section is an example of the steps taken during the design of a simple order entry application. The application is not intended to be complete and workable, but it does provide examples of some of the major design steps and development steps. The steps described are:

- Documenting application functions
- Designing the screens
- Designing the files
- Designing the reports
- Documenting program logic

Design Steps

- Building a development library
- Building a development menu
- Creating development procedures
- Creating screen formats
- Coding the programs
- Testing the programs
- Creating program documentation
- Creating production procedures
- Creating production documentation

Development Steps

DOCUMENTING APPLICATION FUNCTIONS

Ordinarily, the first step after selecting an application is to document the major functions to be performed by the application. In this example, a diagram is used to document the functions. This diagram:

- Identifies operator transactions that will be handled by the application program. A transaction is the exchange of information between the operator and the application program.
- Identifies files that will be used.
- Identifies reports to be generated. In this case, the picking slip is the only report generated.

Figure 5-1 shows the functions to be performed by this order entry application. Notice that screen IDs are assigned to the major screens and that files are identified on the diagram. Also, no effort is made to identify error and exception processing. These items will be considered later as screens are defined and as more detailed program logic is defined.

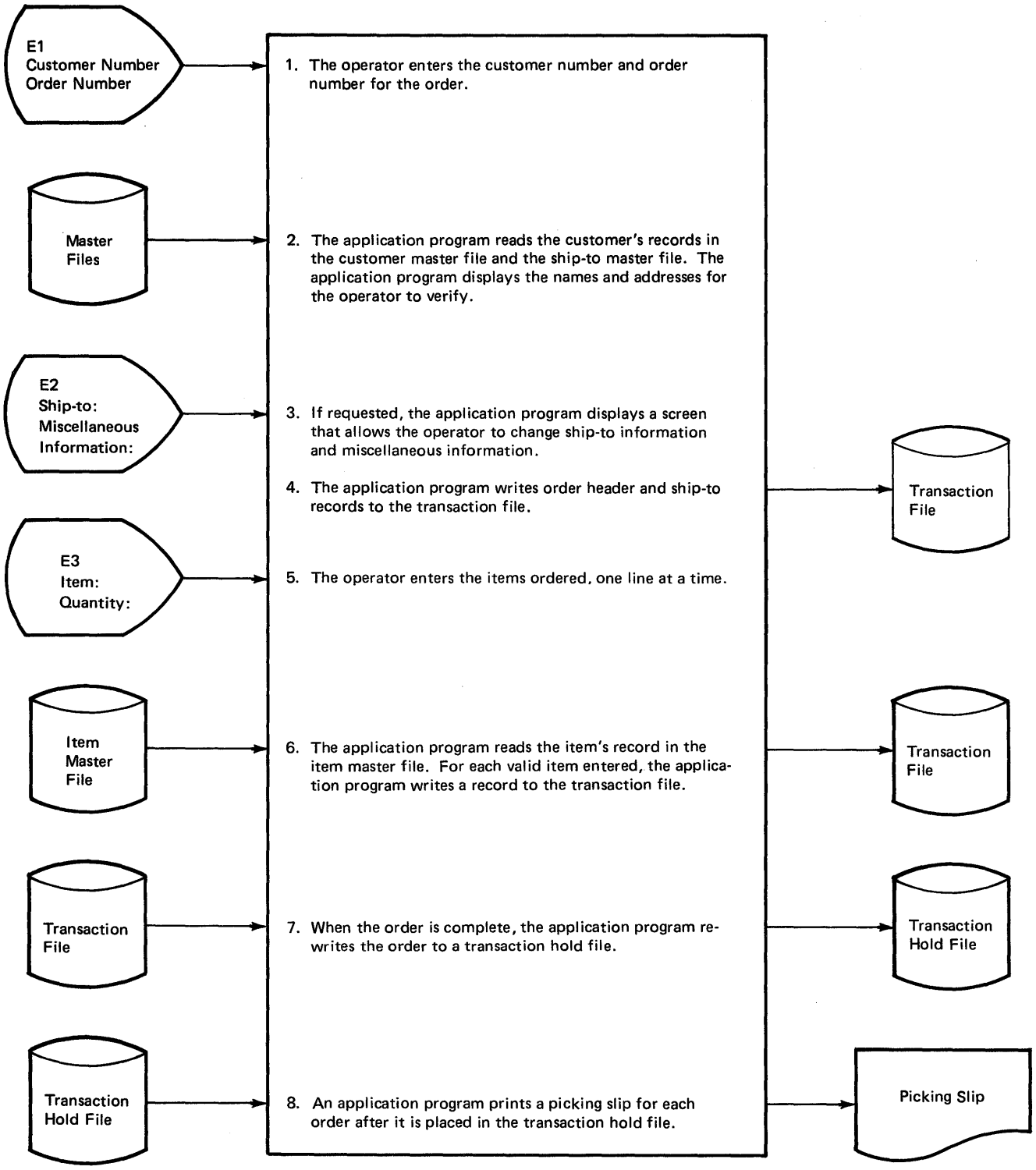


Figure 5-1. Order Entry Application

DESIGNING THE SCREENS

After the major application functions are documented, the screens are usually defined. In this example, the following screen standards are used:

- The first position of a line is usually not used; this allows you to place an attribute character in that position rather than in the last position of the previous line when you use SDA (Screen Design Aid).

- Each screen has a unique screen ID. The first character of the screen ID identifies the application (E indicates order entry), and the second character is a number. The screen ID is the first output/input field on the screen and is a nondisplay field in positions 3 and 4 of line 1. For debugging purposes, the screen ID may be displayed and then changed to a nondisplay field after the program has been tested.

- Screen names are formed by combining a three-character abbreviation of the application, a one-character screen designation, and the two-character screen ID. For example, screen E1 of an order entry application named ORD would be ORDSE1.

- Each screen has a title, centered on line 2 and underlined.

- Each screen has a 48-character error message field on line 23 and/or line 24. Error message text is provided as a constant from within the program. (A 48-character field was chosen to make coding of the RPG II output field easier because a constant of up to 24 characters can be coded on one RPG II output specification.)

- A legend of operator options should be shown in the lower-right corner of the screen. If more space is required, the lower-left portion of the screen (above the error message line) can be used. Command keys are listed in order.

- All constants are displayed with normal intensity.

- All output/input fields are displayed with high intensity.

- When an error is diagnosed, the field in error is displayed in reverse image, and the cursor is positioned at that field. The description of the error condition is displayed on line 23. This error description is also displayed in reverse image. A put override operation is used to display the error screen. The indicator used to request the put override operation is the same indicator used to display the error message, to reverse the image of the field in error, and to position the cursor.

- The screens usually do not instruct the operator to press the Enter/Rec Adv key. The written operator instructions will indicate that the operator should normally press the Enter/Rec Adv key to enter a screen.

- For all input fields on a screen, the operator must press the Field Exit, Field +, or Field - key after entering information in the field.

- Automatic record advance is specified for the last input field on a screen so that the operator need not press the Enter/Rec Adv key if all the fields are entered.

- Numeric fields are right-adjusted and zero filled or right-adjusted and blank filled by specifying a Z or B in the adjust/fill entry on the D specification for the field.

Figure 5-2 shows the form that is used for laying out the screens used by this application. An area is set aside on each sheet for programming notes that apply to the screen. These notes are used when detailed logic of the application programs is defined.

In this example, the screen design process further defines the requirements for the programs. When each screen is designed, the error conditions and exception conditions that can be handled by that screen are identified, and any command keys required to handle those conditions are assigned.

Figure 5-3 shows the layout sheet for screen E1, the first screen. From screen E1, the operator can:

- Key the customer number and order number and request to enter additional information by pressing command key 1.

- Key the customer number and order number and request the screen for entering items ordered by pressing the Enter/Rec Adv key.

- Cancel the order entry process by pressing command key 7.

Figure 5-4 shows the layout sheet for screen E2, which is displayed only if the operator presses command key 1 from display screen E1. Ship-to information and miscellaneous order information can be entered from screen E2.

Figure 5-5 shows the layout sheet for screen E3. From screen E3, the operator enters the individual line items in the order. The operator enters a line item on line 20 of the screen. In this application, lines 13 through 18 show the last six lines entered. In addition to being able to enter a new item, the operator can use this display to step back through the order, to change or delete previously entered line items, or to cancel the order.

Figure 5-6 shows screen E4. Screen E4 is used to display the previously entered lines. Because this screen is displayed using a variable line number and because it is an output screen only, the screen ID is not coded with the screen. The screen ID is still assigned for documentation purposes.

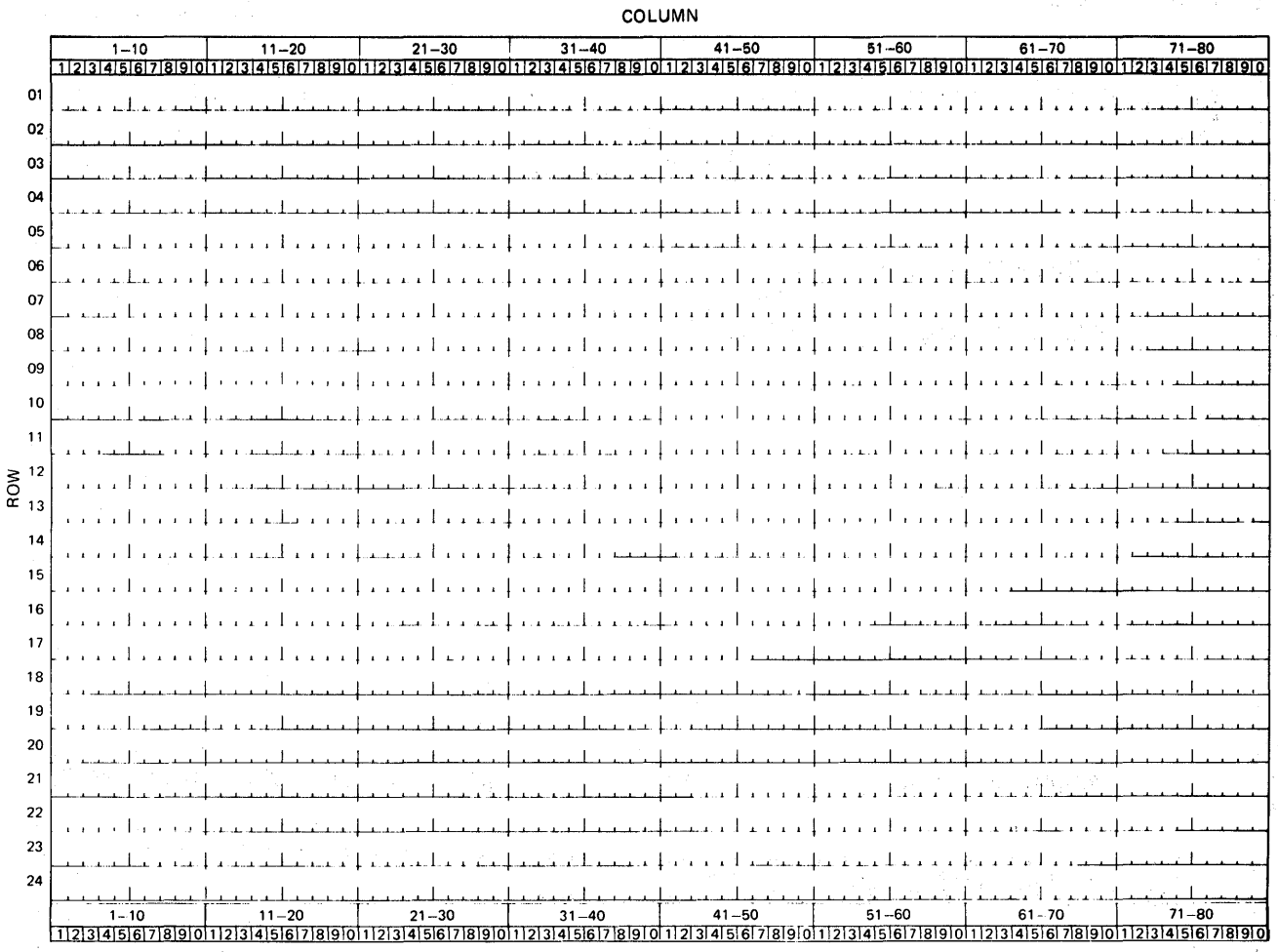
After all the screens are laid out, a list of all the screens, their IDs, and where they are used can be compiled. Figure 5-7 shows such a list for this application.

Screen Name:

Description:

Screen ID:

Display Screen Layout Sheet



Programming Notes:

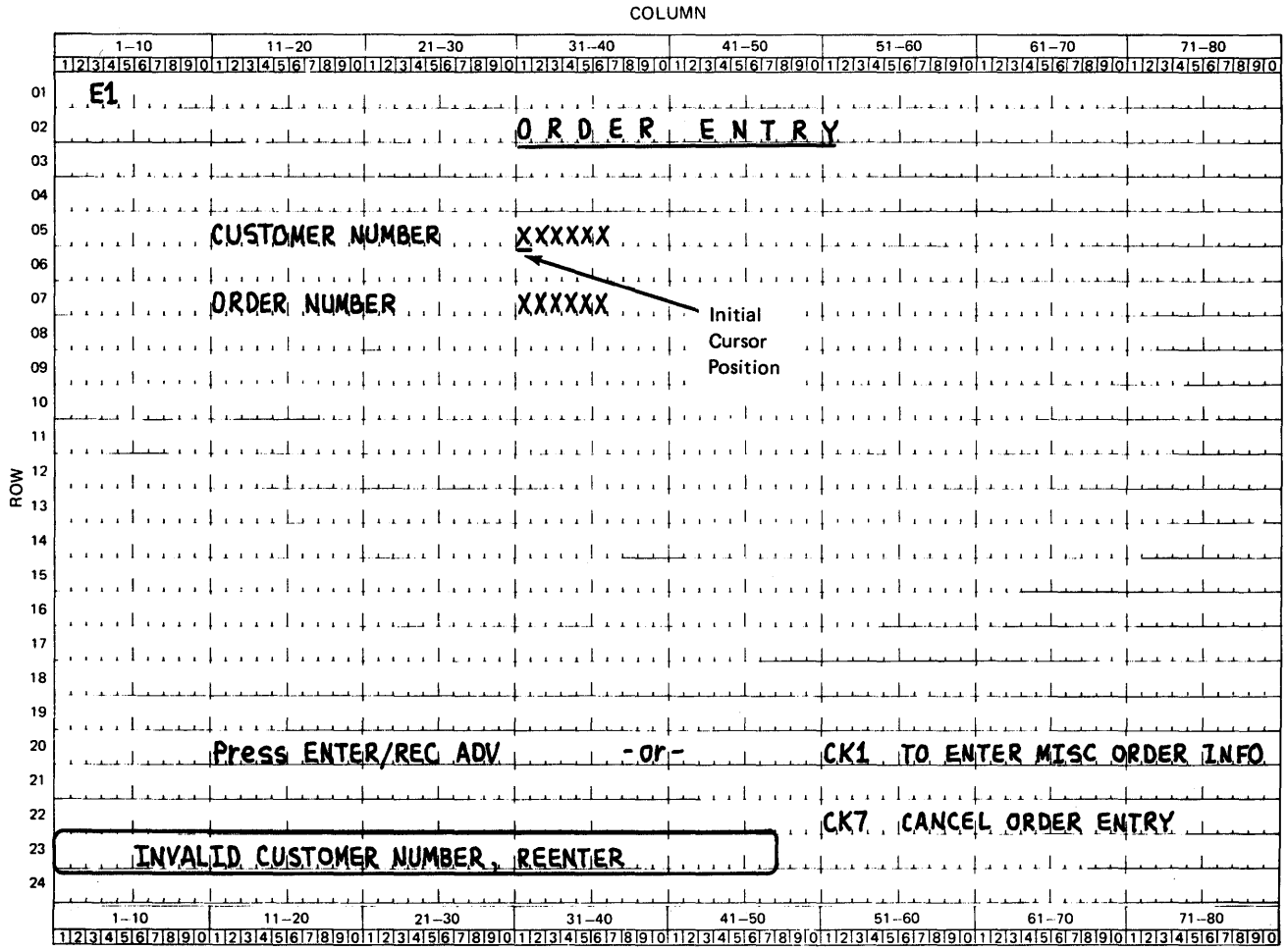
Figure 5-2. Form Used for Laying Out Display Screens

Screen Name: ORDSE1

Description: Order Start Screen

Screen ID: E1

Display Screen Layout Sheet



Programming Notes:

- Other error messages:

  INVALID ORDER NUMBER, REENTER

- The operator should exit from this screen only.
- Control totals could be prompted for here and balanced at the end of the order (this will not be done in this example).
- // PROMPT will be used to display this screen.

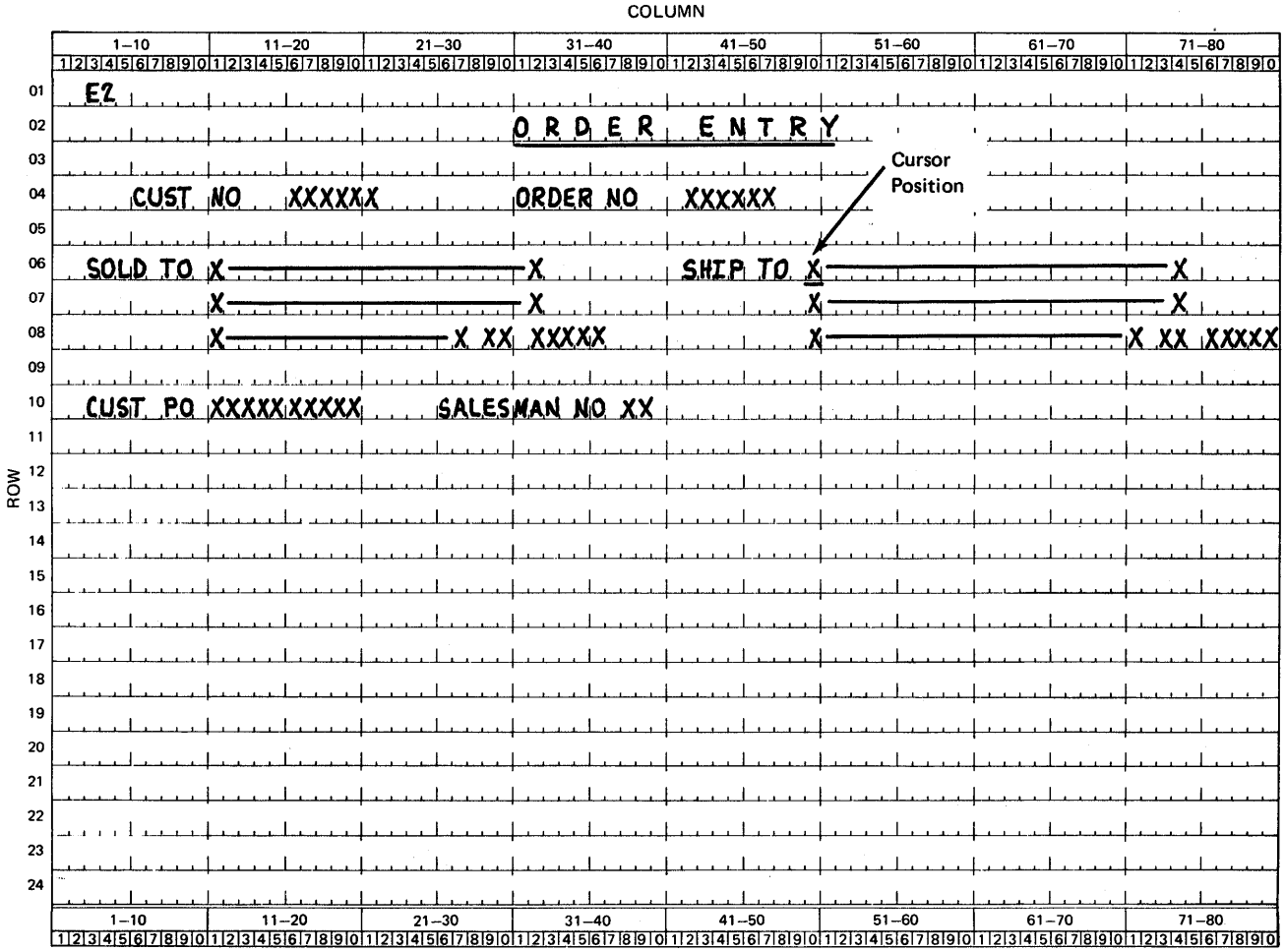
Figure 5-3. Layout of Display Screen E1

Screen Name: ORDSE2

Description: Ship-To or Miscellaneous Information Entry

Screen ID: E2

Display Screen Layout Sheet



Programming Notes:

- This screen appears if CK1 was pressed when screen E1 was displayed.
- Ship-to information will be displayed if a ship-to record was read. The operator can override the information.

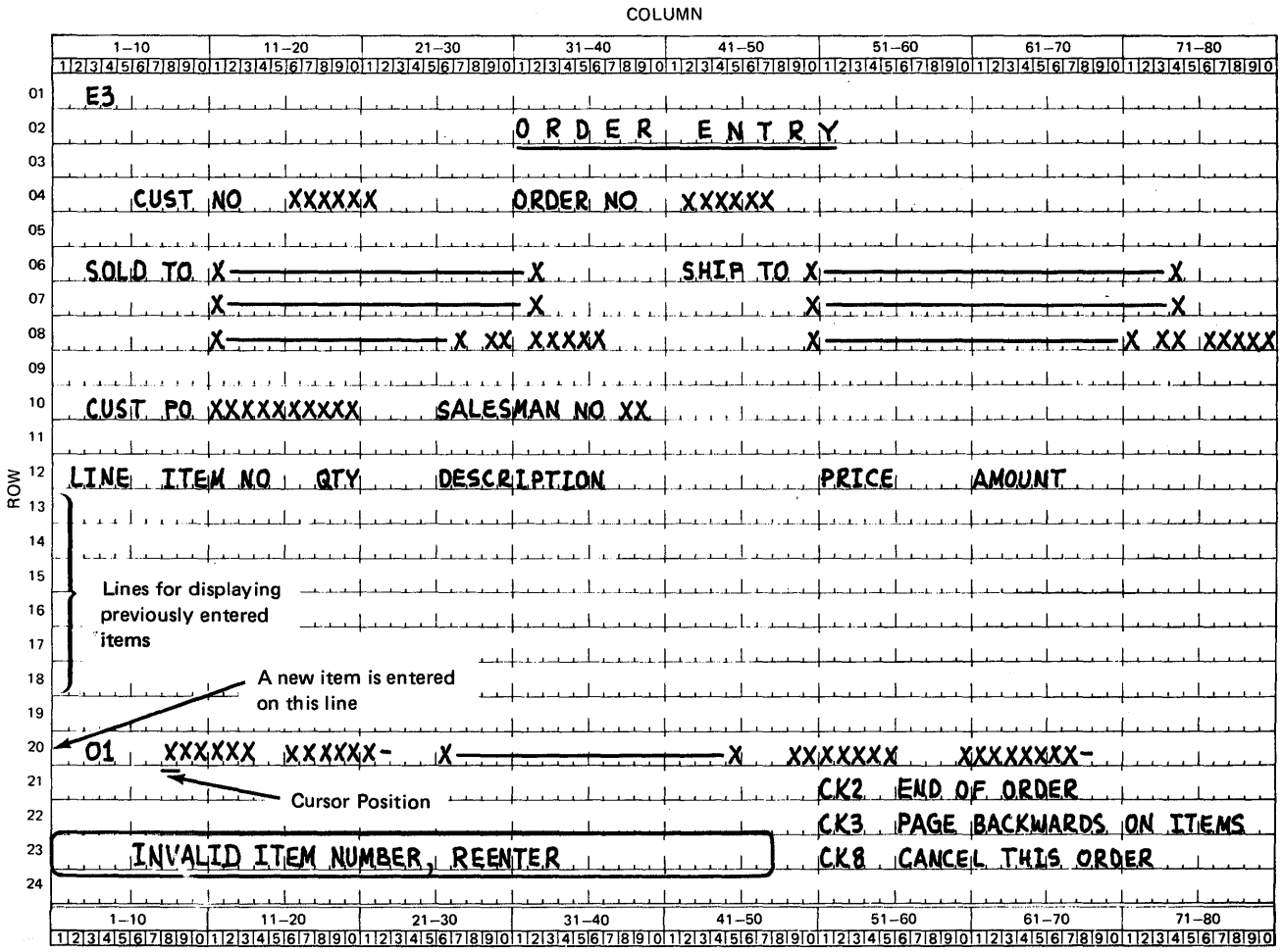
Figure 5-4. Layout of Display Screen E2

Screen Name: ORDSE3

Description: Line Item Entry

Screen ID: E3

Display Screen Layout Sheet



Programming Notes:

- This screen appears (1) if the Enter/Rec Adv key was pressed when screen E1 was displayed, or (2) after screen E2 appears.
- Field backspace should allow the line number to be entered (used to change a line item).
- If only a line number is entered, the program should delete the line item.
- Lines 13-18 are reserved to display up to six previously entered items.
- Item number and quantity can only be entered for new line items.

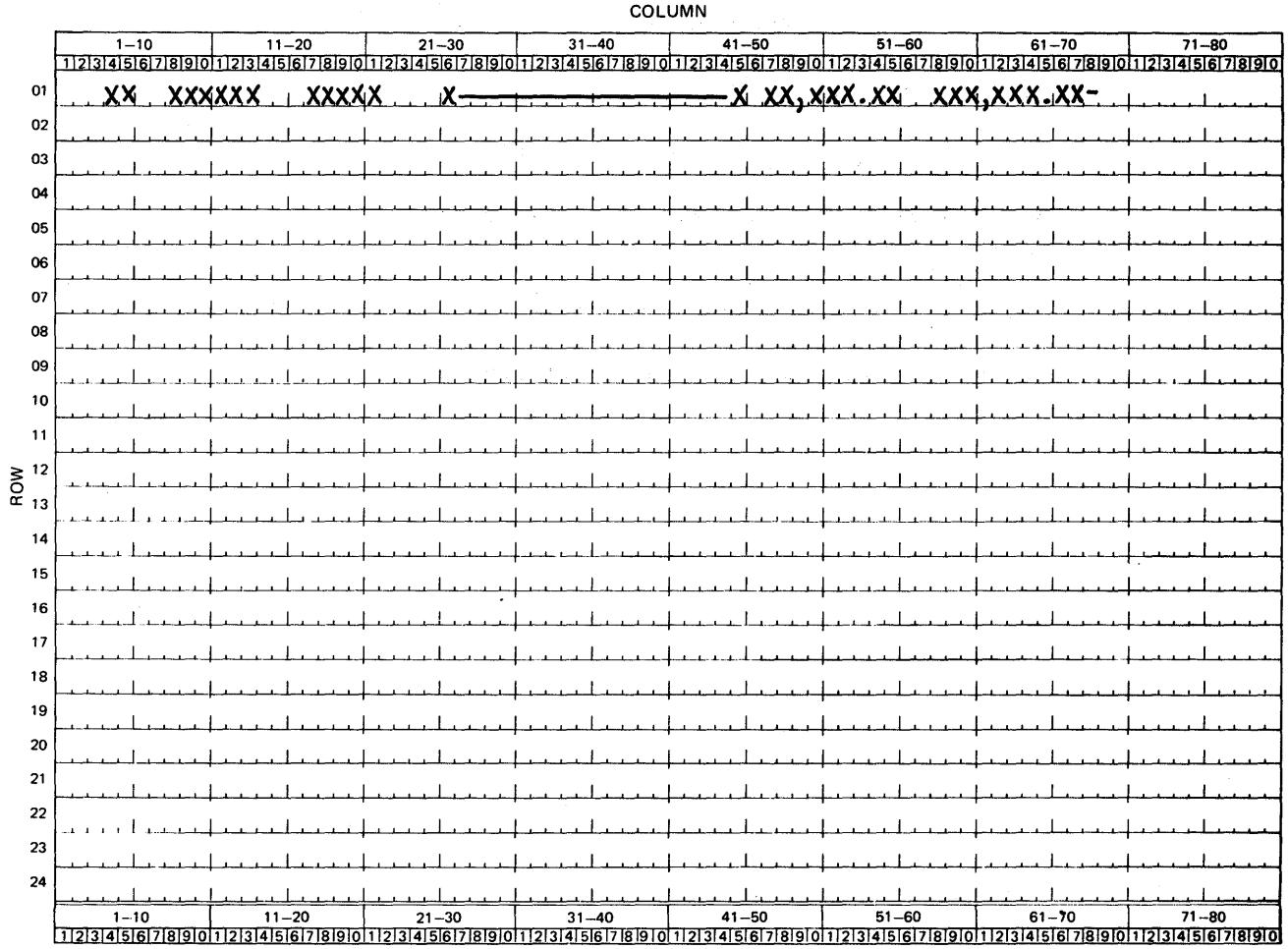
Figure 5-5. Layout of Display Screen E3

Screen Name: ORDSE4

Description: Entered Items Display

Screen ID: E4

Display Screen Layout Sheet



Programming Notes:

- Fields are displayed only after editing is done.
- Up to 6 lines will be displayed.
- Variable line numbers will be used - start on line 13.

Figure 5-6. Layout of Display Screen E4

Screen ID	Screen Name	Where Used
E1	ORDSE1	Order entry start
E2	ORDSE2	Ship-to override and miscellaneous information
E3	ORDSE3	Line item entry
E4	ORDSE4	Line item display (on variable line)

Note: The screen IDs are used in the input specifications for the RPG II program to identify the screen being read and to turn on an input indicator. The screen names are used in the S specifications for the screens and in the output specifications for the RPG II program to display the desired screen.

Figure 5-7. List of Display Screens Used in the Order Entry Example

DESIGNING DISK FILES

At this point, disk file requirements are defined and new files designed.

Master Files

In this example, three master files are used:

- CMAST–the customer master file
- SMAST–the ship-to master file
- IMAST–the item master file

For each of these files, the fields in the file are listed along with a field description and the field length. Alphameric fields are denoted by an A, and numeric fields are denoted by an N. The from and to locations and the field names are assigned. Figure 5-8 shows the list of the fields in each of the master files. Meaningful field names are used when possible. The first character of the name identifies the file.

Notes:

1. The record layouts do not show all the fields that would actually be required in an order entry/billing application. For example, no on-hand quantities are shown for inventory control.
2. Numeric fields are not packed in this example so that the example will be easier to follow. In a real application, numeric fields are normally packed (with two digits stored in each byte except the rightmost byte) to conserve disk space.

File name: CMAST
File organization: Indexed
Key: Customer number (CUSNO)
Record length: 128

Field Description	Length	Decimal Position	Data Format	Location From	To	Field Name
Record code–MA	2		A	1	2	CRECCD
Delete code–D (blank if not active)	1		A	3	3	CDELETE
Customer number	6	0	N	4	9	CUSNO
Customer name	25		A	10	34	CNAME
Customer address	25		A	35	59	CADDR
City	22		A	60	81	CCITY
State	2		A	82	83	CSTATE
Zip code	5	0	N	84	88	CZIPCD
Salesman number	2	0	N	89	90	CSLSNO

File name: SMAST
File organization: Indexed
Key: Customer number (SCUSNO)
Record length: 128

Field Description	Length	Decimal Position	Data Format	Location From	To	Field Name
Record code–SA	2		A	1	2	SRECCD
Delete code–D (blank if not active)	1		A	3	3	SDELET
Customer number	6	0	N	4	9	SCUSNO
Customer name	25		A	10	34	SNAME
Customer address	25		A	35	59	SADDR
City	22		A	60	81	SCITY
State	2		A	82	83	SSTATE
Zip code	5	0	N	84	88	SZIPCD

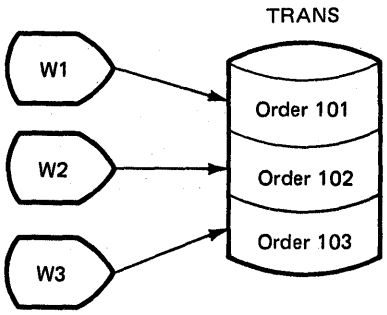
File name: IMAST
File organization: Indexed
Key: Item number (ITEMNO)
Record length: 128

Field Description	Length	Decimal Position	Data Format	Location From	To	Field Name
Record code–IT	2		A	1	2	IRECCD
Delete code–D (blank if not active)	1		A	3	3	IDELET
Item number	6	0	N	4	9	ITEMNO
Description	20		A	10	29	IDESCR
Price	6	2	N	30	33	IPRICE
Warehouse location	5		A	34	38	IWHLOC

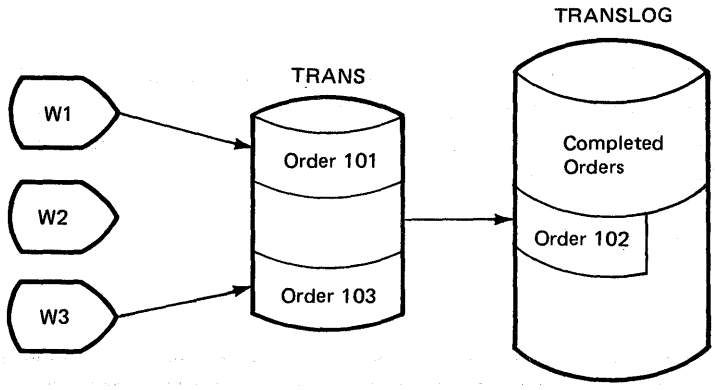
Figure 5-8. Record Layouts for Master Files

Transaction Files

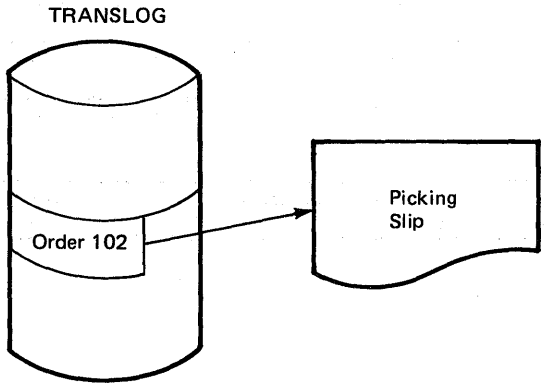
For this application, two transaction files are used. As an order is being built, it is placed in a partitioned direct file called TRANS. Because, in the example, a maximum of three display stations can be entering orders at one time, the file has three partitions. Each partition is assigned for the use of one of the three possible display stations.



After an order has been completely entered, the order is copied from the TRANS file into a file that contains all orders that have been entered. That file is called TRANSLOG. After the order has been copied from TRANS, the partition that the order occupied can be used for another order. In the following diagram, the operator at display station W2 completes an order; that order is then written to the TRANSLOG file.



The picking slip is printed from the information in the TRANSLOG file.



By using the intermediate partitioned file, the designer can achieve the faster response times that normally result from direct file processing. Also, a simple access algorithm is used because, after a record is written to the TRANS file, the program simply increments the relative record number by one to determine the location of the next record in the order. Using the two transaction files requires less disk space than if one partitioned direct file were used. If one partitioned transaction file had been used, each partition in that file would have to be large enough to hold the maximum number of orders that can be entered from any display station.

Note: An alternative to using a partitioned direct transaction file would be to have a different transaction file for each user or display station. The unique file names could be formed by appending the user ID or work station ID to the file name (for example, FILE?WS?). The technique of using multiple transaction files normally works well if the data entry program is an SRT program.

The following paragraphs briefly describe the file layout and record formats for the transaction file.

TRANS File

TRANS file organization:

1	100 101	200 201	300 301
Area used by display station 1	Area used by display station 2	Area used by display station 3	Control Record

Note: If orders can be entered from several display stations, consider making the first record of each partition a control record. This reduces the sector contention/lockout that might occur when several users require the control record at the same time.

Records in TRANS file:

128 Bytes

Record 1	CU		Order Number	Customer Number	Customer Name	Address 1		Address 2		Zip	
2	CS		"	"	Ship-to Name	Address 1		Address 2		Zip	
3	IT		"	"	Item Number	Line No.	Description	Quantity	Price	Amount	
4	IT		"	"	Item Number	Line No.	Description	Quantity	Price	Amount	
.											
.											
.											
301	Relative record number of last record used in partition 1				Relative record number of last record used in partition 2			Relative record number of last record used in partition 3			

The transaction file contains three different record types:

- The customer record, which is identified by a CU in the first two positions. The customer record contains the order number, the customer number, the customer's address, the salesman's number, and the purchase order number.

- The ship-to record, which is identified by a CS in the first two positions. This type of record is optional.

- The item records, which are identified by an IT in the first two positions. An item record exists for each line item entered by the operator.

All records in the file must be the same length. In this example, the records are 128 bytes long. Because the records must all be the same length, some disk space is wasted in each record. If the space required for different record types differs greatly, you might want to break the longer record into segments or place the different record types into separate files.

Figure 5-9 shows the record layout for the TRANS file.

File name: TRANS
File organization: Direct
Record length: 128

Field Description	Length	Decimal Position	Data Format	Location From	To	Field Name
Record code–CU	2		A	1	2	OCODE
Delete code–D (blank if not active)	1		A	3	3	ODELET
Customer number	6	0	N	4	9	CUSNO
Order number	6	0	N	10	15	ORDNO
Customer name	25		A	16	40	CNAME
Customer address	25		A	41	65	CADDR
City	22			66	87	CCITY
State	2		A	88	89	CSTATE
Zip code	5	0	N	90	94	CZIPCD
Salesman number	2	0	N	95	96	CSLSNO
Purchase order number	10		A	97	106	CPONO
Ship-to record:						
Record code–CS	2		A	1	2	OCODE
Delete code–D (blank if not active)	1		A	3	3	ODELET
Customer number	6	0	N	4	9	CUSNO
Order number	6	0	N	10	15	ORDNO
Ship-to name	25		A	16	40	SNAME
Ship-to address	25		A	41	65	SADDR
City	22		A	66	87	SCITY
State	2		A	88	89	SSTATE
Zip code	5	0	N	90	94	SZIPCD
Line item record:						
Record code–IT	2		A	1	2	OCODE
Delete code–D (blank if not active)	1		A	3	3	ODELET
Customer number	6	0	N	4	9	CUSNO
Order number	6	0	N	10	15	ORDNO
Order line number	2	0	N	16	17	OLINE
Item number	6	0	N	18	23	ITMNO
Item description	20		A	24	43	IDESCR
Quantity ordered	6	0	N	44	49	OQTY
Price	6	2	N	50	55	OPRICE
Amount extended	8	2	N	56	63	OAMT
Warehouse location	5		A	64	68	IWHLOC

Figure 5-9 (Part 1 of 2). Record Layout of TRANS File

Field Description	Length	Decimal Position	Data Format	Location From	To	Field Name
Control record:						
Record code–QU	2		A	1	2	RCODE
Relative record Number Used by Display Station 1	3	0	N	3	5	RR#1
Relative record Number Used by Display Station 2	3	0	N	6	8	RR#2
Relative record Number Used by Display Station 3	3	0	N	9	11	RR#3

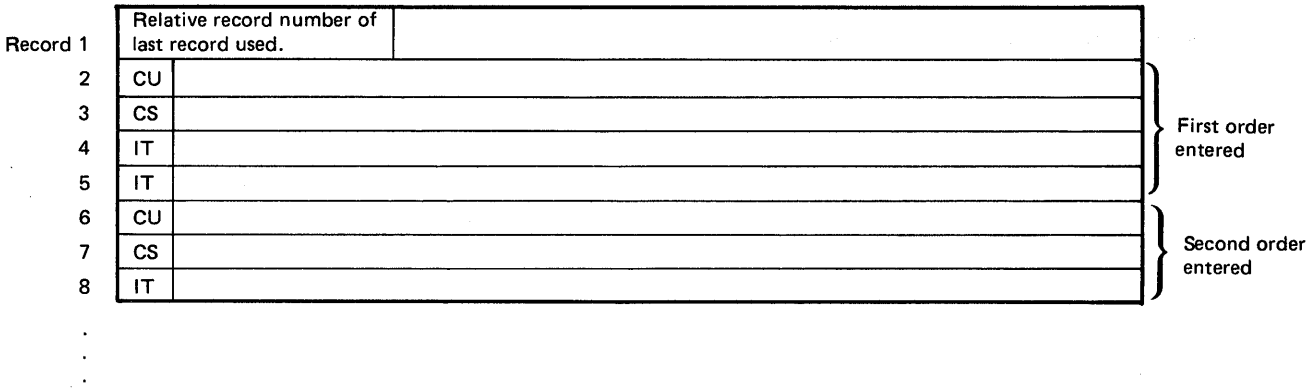
Figure 5-9 (Part 2 of 2). Record Layout of TRANS File

TRANSLOG File

For this example, the TRANSLOG file is also a direct file. The control record, which is the first record in the file, identifies the last record used in the file. Orders are placed in the file as they are completed by each display station operator. The first record of an order immediately follows the last record of the previous order written to the file.

Note: If inquiry programs use this file, an indexed organization would probably be better. The record key could be the order number plus the line number. The inquiry program could locate an order by chaining to the first record of the order. If the order file was a direct file, an inquiry program would have to search for the first relative record number of the requested order.

TRANSLOG file organization:



The record layout is the same as for the TRANS file except for the control record.

DESIGNING THE REPORT

Figure 5-10 shows the picking slip, which is printed for each order. The top inch of the form is reserved for the company's name and address, the phone number, report title and number, date and page number, and any instructions to the customer.

The printed lines are arranged so that as few lines as possible are printed. Because this form is used by warehouse personnel to pick the order, some information has to be entered by the picker. Examples of information entered by the picker are PICKED BY, DATE, and QUANTITY SHIPPED. The form should be long enough to list the average number of line items in an order.

Note: If warehouse locations are used, the line items could be listed in order by location. This type of listing requires an intermediate sort step performed by either the sort utility program or a user-written program. The example does not sort the line items by warehouse location.

Considerations for Designing Output Reports

Certain considerations to be taken into account when you are designing the output report:

- Leave enough space on the left edge of the report in case you have to bind the report.
- Separate each field on the report by at least one space from an adjacent field.
- Group information items that are similar.
- Number all pages of a report. By using page numbers you can have the operator restart the printing of a job from a specified page number instead of the beginning of the job in the event that you have to restart a job that was printing a report. To restart the printing of a job on a page number other than page one you should use print spooling or code the restart capability in your program.
- Provide meaningful headings for the data on the report. Abbreviations, codes, and special symbols should be avoided. If you need more space than is available on the output design sheet, use several lines for long headings, rather than abbreviating them.
- Remember that the appearance of the report is important. The report should be easy to read and self-explanatory.

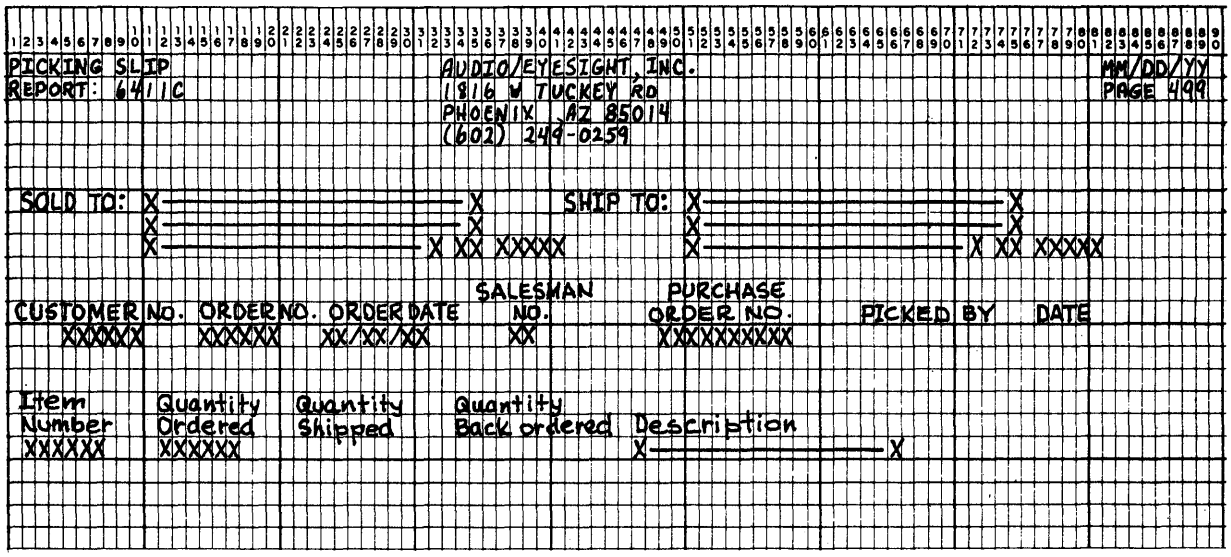


Figure 5-10. Picking Slip for Order Entry Application

Determining Program Requirements

After the screens are designed, the program requirements can be further identified. For this application, a transaction-oriented approach is used. Each program processes a limited type of operator transactions.

Program names are formed by combining a three-character application abbreviation and a three-character function abbreviation. For example, ORDHDR is a program from the order entry application, and this program processes header records.

Program	Function
ORDHDR	Processes the screen on which the operator enters the customer number and the order number. This program also allows the operator to change miscellaneous information about the order.
ORDITM	Processes each line item after it is entered.
ORDPRT	Prints a picking slip for each order after it is entered.

After the program requirements are determined, some basic design decisions can be made. In this example, the following decisions are made:

- Keying by the operator should overlap program initiation. For ORDHDR, a // PROMPT statement will be used to display screen E1 before the ORDHDR program is loaded. Before ORDHDR ends, it will display screen E3 so that the operator can be keying the first order line while ORDITM is being initiated.

- ORDHDR will be an SRT program. The SRT attribute is selected because:
  - Data entry overlaps program initiation; therefore, initiation time does not significantly affect response time.
  - The program is in main storage for a relatively short time.
  - All three operators will probably not initiate an order at the same time.

- ORDITM will be an MRT program. The MRT attribute was selected mainly because the program is in storage for a long time and because all three of the operators will often be entering line items at the same time.

- ORDPRT will be a no-requestor-terminal (NRT) program. The NRT attribute was chosen because ORDPRT is a resource-handling program that requires no operator interaction.

After the program requirements are identified and some of the basic design decisions are made, the logic of the programs can be defined and documented. In this case, the logic documentation consists of a flowchart and a written program description that identifies:

- Data to be keyed

- Data to be sent from the program to the display station

- Required disk accesses by type (input, output, or update)

- Editing to be done

- Options available to the operator

- Calculations to be performed

Figures 5-11 through 5-13 show the flowcharts and program descriptions for the programs in this example.

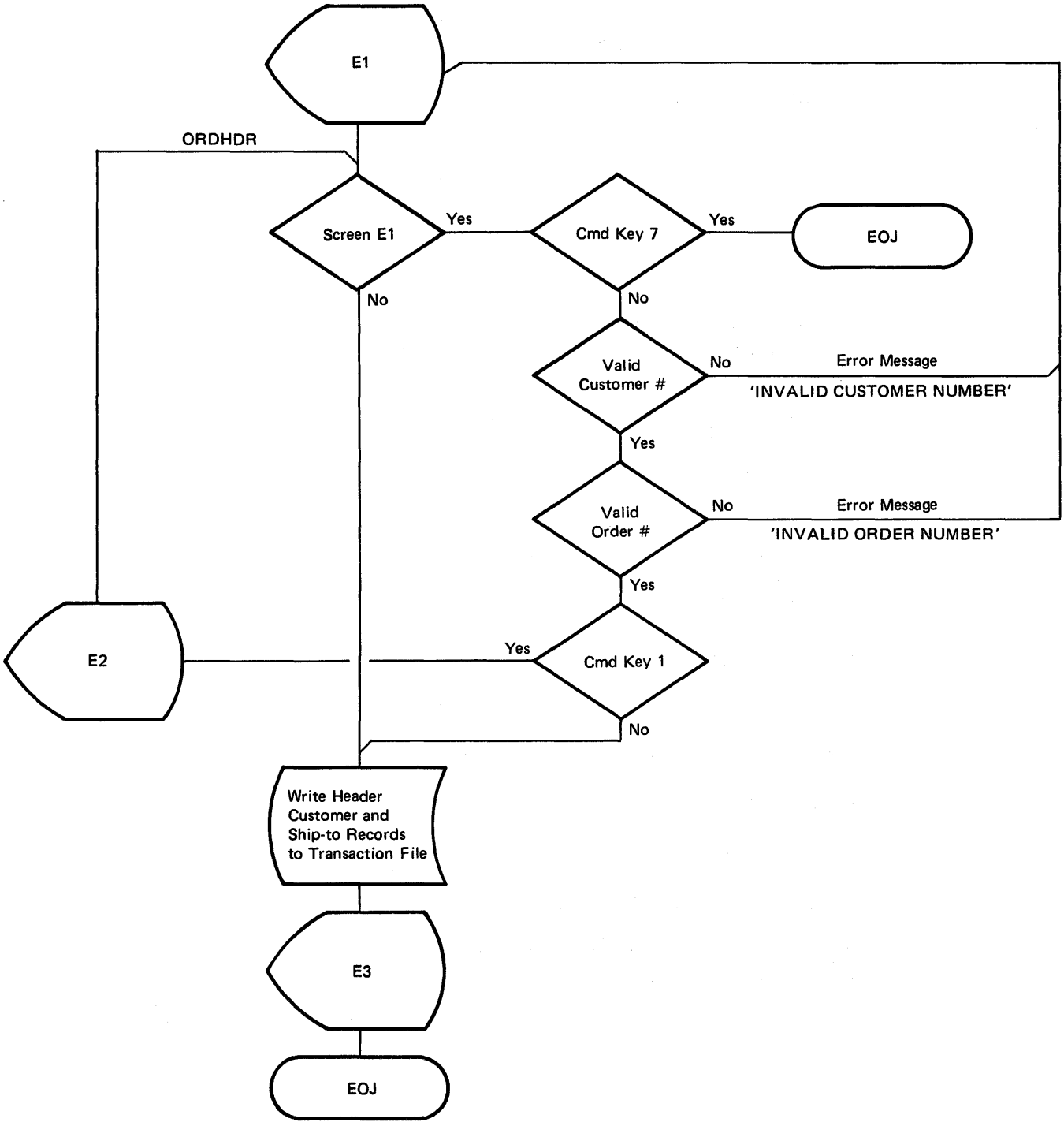


Figure 5-11 (Part 1 of 2). Logic Documentation for ORDHDR

ORDHDR Program Description

ORDHDR is an RPG II SRT program that performs the following functions:

1. Accept input from either screen E1 or E2.

2. Enter the customer number and order number on screen E1.

3. If command key is pressed (KG indicator on), set on external switch U1 and LR indicators. Because the program ends, skip the rest of the calculations.

4. From screen E1:
   a. Chain to the customer master file, CMAST, to read the customer's record and also chain to the ship-to file, SMAST.
   b. If the customer's record is not found, write the 'INVALID CUSTOMER NUMBER, REENTER' error message to screen E1.
   c. If the order number is blank, write the 'INVALID ORDER NUMBER, REENTER' error message to screen E1.
   d. If the customer number and the order number are valid and if command key 1 is pressed (KA ison), display screen E2.

5. From screen E2, accept ship-to override and/or miscellaneous order information.

6. Write to the TRANS file by:
   a. Using a table lookup to find which partition to use. For example:
      Use an array to read three relative record numbers, RR1, RR2, and RR3.

TABWS	TABP
W1	1
W2	2
W3	3

   b. Chaining to the control record, relative record number 301.
   c. If the corresponding last used relative record number (RR#,I) is blank, Z-ADD (I x 100 + 1). If it is not blank, use RR#,I+1 to write the customer order header, CU record code, including the user ID.
   d. If screen E2 was used, chain to RR#,I+1 and write a second record, the ship-to and miscellaneous record, CS record code.
   e. Chain back to the control record and update the RR# array.

7. Set up the line number for the items by initializing it to 1.

8. Display screen E3 showing customer's name and address, ship-to, miscellaneous information and headings for item entry.

9. Set on LR.

Figure 5-11 (Part 2 of 2). Logic Documentation for ORDHDR

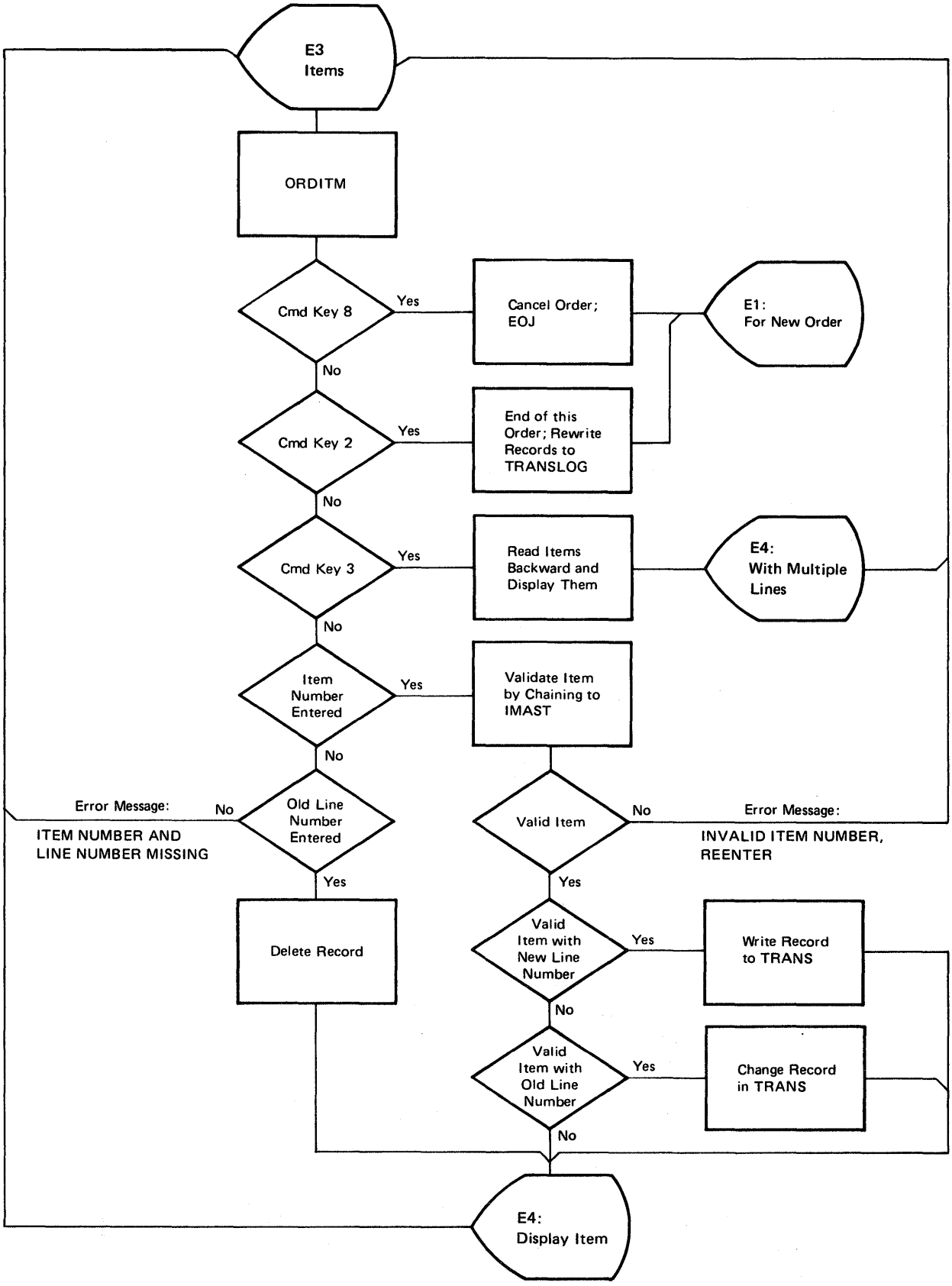


Figure 5-12 (Part 1 of 2). Logic Documentation for ORDITM

ORDITM Program Description

ORDITM is an RPG II MRT program that performs the following functions:

1. Accept input from screen E3.

2. The item number and quantity are the minimum information to be entered.

3. Save the work station ID and indicators. Set up a field for the variable line number and code the program for three display stations.

4. If command key 8 is pressed (KH is on), the current order being worked on should be canceled. Reset to zero the corresponding relative record number in the TRANS file for the display station. The order will not be written to the TRANSLOG file. Display screen E1 to enter another order.

5. If command key 2 is pressed (KB is on), the order is complete. Chain to the control record of the TRANS file to find out how many records have to be copied to the TRANSLOG file. Chain to the control record (the first record) of the TRANSLOG file to find the last relative record used. Read a record from TRANS and write the record to TRANSLOG until all records for the order have been copied. At the end, update the control record of TRANSLOG. Display screen E1 to enter another order.

6. If command key 3 is pressed (KC is on), read the control record of TRANS and read backwards and display one at a time using screen E4. Display a maximum of six lines at a time. Then command key 3 can again be pressed. Save the last relative record number displayed.

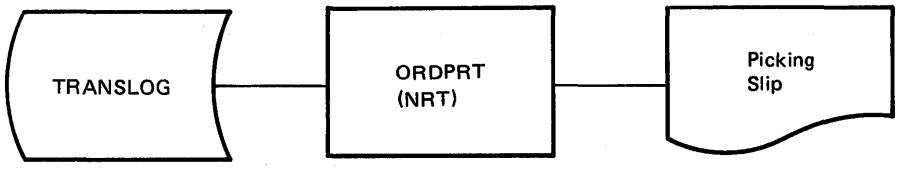
7. If an item number is entered, chain to the item master file, IMAST, to read the item's record. If the item record is not found, write the 'INVALID ITEM NUMBER, REENTER' error message to screen E3. If the item is found, write the record to the TRANS file, IT record code, and update the control record of the TRANS file.

8. If a line number less than the current one is entered, a record will be changed or deleted. If only the line number is entered, the line item is deleted. Read the previously entered items backwards from the TRANS file and tag the one to be deleted by placing a D in position 3.

9. Add one to the line number to indicate the next item to be entered. Display the previously entered line on screen E4. Then display screen E3 to accept the next line item. Be sure to check the variable line's value. Its initial value is 13. Add one for each line displayed until 18 is reached, then reset to 13.

Figure 5-12 (Part 2 of 2). Logic Documentation for ORDITM

Picking Slip Program



ORDPRT Program Description

ORDPRT is an RPG II NRT program that performs the following functions:

1. Read the control record of the TRANSLOG file to find the last record printed.
2. Read each record for the order and print each line.
3. At the end of the order, update the control record with the relative record number of the last record printed.

Figure 5-13. Logic Documentation for ORDPRT

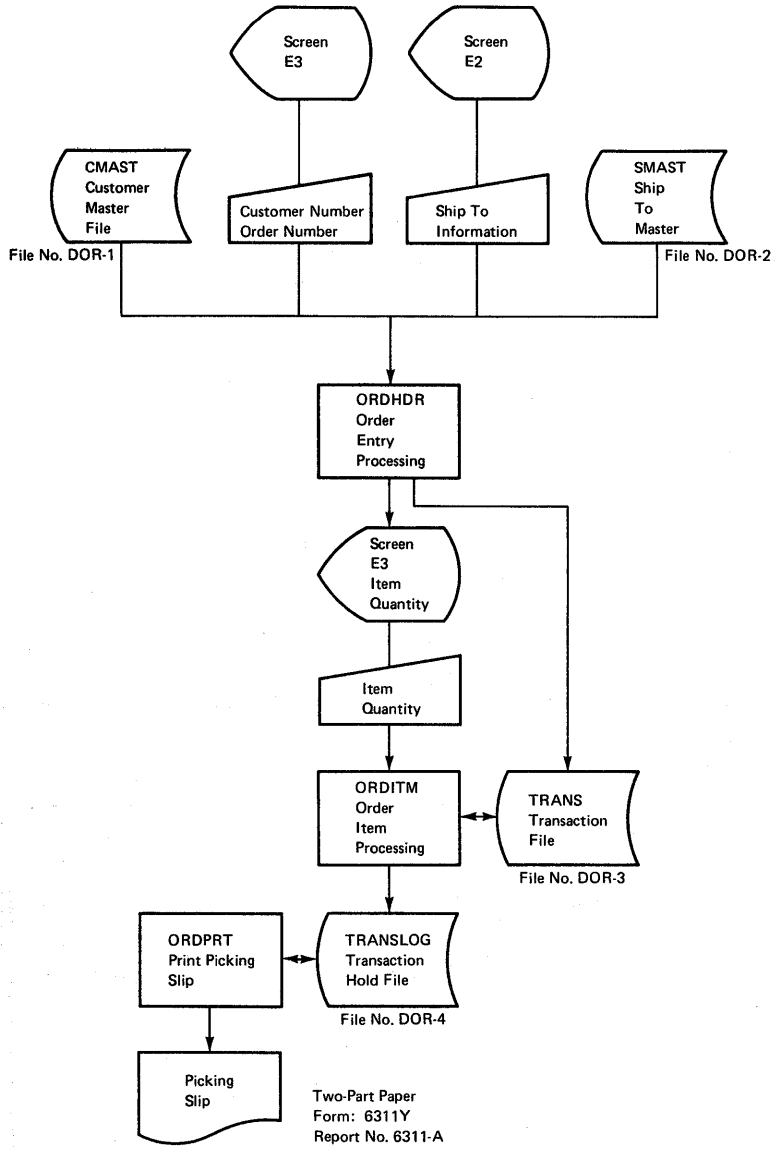
System Flowchart

After you have completed the requirements definitions and documented the logic needed in your programs, you should prepare a general systems flow of your application. The system flowchart should identify the basic operations of your programs and the resources needed by your programs, and should include any information you find helpful in explaining the application. The system flowchart is used to show the basic inputs, outputs, processing steps, and processing programs.

The following diagram is a sample of a system flowchart for the order entry application.

TITLE ORDER ENTRY APPLICATION

ORG. Programming	DATE 01/15/78
AUTHOR A. Programmer	PAGE 1 OF 1



Notes

Run Frequency: Daily
Volume: Approximately 500 transactions per day

ORDHDR (OR-01)

Program Type: SRT
Est Elapsed Time: 1 hour

Accepts customer number, order number, and ship-to information, and writes header customer and ship-to-records to the TRANSACTION file.

ORDITM (OR-02)

Program Type: MRT
Est Elapsed Time: 30 minutes

Accepts item number and quantity information, and writes an order record to the TRANSLOG file.

ORDPRT (OR-03)

Program Type: NRT
Est Elapsed Time: 10 minutes

Generates picking slips, and updates a control record in the TRANSLOG file.

BUILDING A DEVELOPMENT LIBRARY

Often, the first step in application development is to build a development library. For this example a development library called TESTLIB can be built by entering the following procedure command:

```
BLDLIBR TESTLIB,100,20
```

BUILDING A DEVELOPMENT MENU

After the development library is built, a development menu can be created. A development menu saves much time by allowing the developer to select often-used functions, such as modifying and compiling an RPG II program, from a menu.

To build a development menu called TESTM and store it in TESTLIB, enter the following SDA command:

```
SDA TESTM,TESTLIB
```

For detailed information about how you can use SDA to interactively build menus, refer to the *SDA Reference Manual.*

The development menu for this example is shown below. In this case, the developer has taken advantage of the freedom of design that free-format menus provide by grouping related functions and separating those groups with dashed lines.

```
COMMAND                                                          W7
                                  MENU: TESTM
:--------------------------------:-----------------------------------------:
: SDA      1. Screen Design      :  MENU    13. System functions           :
:          2. Menu Build         :          14. Other library functions    :
:--------------------------------:-----------------------------------------:
: SEU      3. Procedure          :                                         :
:          4. Program            :  MISC    15. Initialize diskette        :
:--------------------------------:          16. Backup library             :
: COMPILE  5. RPG II             :          17. Display file records       :
:          6. WSU                :          18. Remove library members     :
:--------------------------------:          19. Data File Utility          :
: CATALOG  7. Disk               :          20. BASIC                      :
:          8. Diskette           :-----------------------------------------:
:--------------------------------:  LIST    21. Procedure                  :
: FILE     9. Save               :          22. Program                    :
:         10. Restore            :          23. Directory                  :
:         11. Delete             :-----------------------------------------:
:         12. Rename/Reorganize  :  CHANGE  24. Change Session library/menu:
:--------------------------------:-----------------------------------------:
ENTER NUMBER, COMMAND, OR OCL.


_                                                          <- READY
```

CREATING DEVELOPMENT PROCEDURES

After building the menu, you can use SEU to create each of the procedures used by the TESTM menu. To create each procedure, you could enter the following command:

```
SEU name,P,,,TESTLIB
```

where name is the name of the procedure being created. Or you can use item 3, once it is created, from the TESTM procedure. The following sections describe the procedures for three of the functions on the development menu. The procedures described are:

- Using SEU to update and recompile a program (item 4 on the menu)
- Saving disk files (item 9 on the menu)
- Changing the session library and/or menu (item 24 on the menu)

These procedures show many of the functions that can be incorporated into procedures to make them easier to use. For further information about coding procedures, refer to Chapter 5 of the *SSP Reference Manual*.

Following the descriptions of the procedures is a section that lists the screen format specifications for the PROMPT screens displayed by the procedures and lists the contents of the procedures.

Using SEU to Update and Recompile a Program (ZSEUR)

A procedure called ZSEUR is called when menu item 4 is selected. The procedure intially displays a format that prompts for the source program name. The current session library is also displayed, but the cursor is positioned at the source program name.

```
                    SEU UPDATE OF RPG II PROGRAM


Current session library      --->     TESTLIB

Source member name           --->     PROGA

Is member an RPG II program  --->     Y
```

The RPG II compiler is not called unless the Y is entered.

The ZSEUR procedure then calls SEU execution so the user can create or update the program or source member. After SEU execution ends, ZSEUR displays another PROMPT screen from which the user can compile the program.

```
                    COMPILE OPTION OF RPG II PROGRAM


          Current session library      --->    TESTLIB

          Source program name          --->    PRDGA

          Standard default compile     --->    YES

          Run from JOBQ  (0-no,1-yes) --->    1

          XREF list req  (0-no,1-yes) --->    0

(Standard Default parameters:
 Replace assumed, source and load processed from current
 session library.  60 Blocks used for work files.)
```

If the user leaves the third field as is (YES), the program is compiled using default parameters defined within the ZSEUR procedure. If the user enters another value (for instance, NO) in the third field, the procedure calls the RPG II prompt screen.

In summary, the ZSEUR procedure helps with the typical development task of changing and recompiling a program by combining the two steps into one procedure. The specifications for the PROMPT screens and the contents of the ZSEUR procedure are in PROMPT screens and the contents of the ZSEUR procedure are in *Listings for Sample Development Procedures*, which follows the description of the ZSAVEF and ZLIBCHNG procedures.

Saving Disk Files (ZSAVEF)

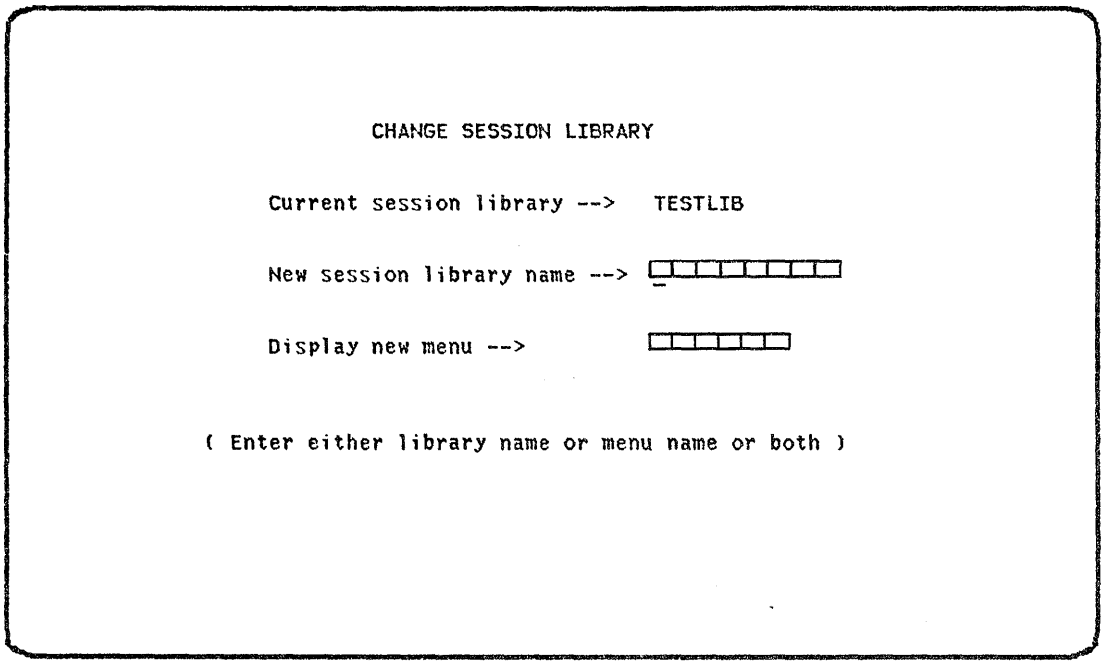
A procedure called ZSAVEF is called when menu item 9 is selected. ZSAVEF displays the following prompt screen, which allows the user to copy all files in a file group or to copy up to five individual files. The retention days of 999, volume ID of IBMIRD, and location of S1 are coded as a default, but they can be changed.

```
                        SAVE FILE OPTIONS

Enter:   retention days                  (1 to 999) --->  999

         date if more than one file exists          --->

         volume-ID of diskette                      --->  IBMIRD

         location          (S1,S2,S3,M1.nn,M2.nn)   --->  S1

Enter group name if desired:                        --->

Enter up to 5 file names:                           --->  _

   - all files on diskette                   (ALL) --->

   - an individual file                     (name) --->

                                                    --->

                                                    --->
```

The ZSAVEF procedure allows the user to save more than one file on diskette, unlike the SAVE procedure (described in the *SSP Reference Manual*), which can only save one. The specifications for the prompt screen and the ZSAVEF procedure are in *Listings for Sample Development Procedures*, which follows the description of the ZLIBCHNG procedure.

Changing the Session Library and/or Menu (ZLIBCHNG)

A procedure called ZLIBCHNG is called when menu item 24 is selected. The ZLIBCHNG procedure displays the current library and allows the user to change the session library and/or the current menu. The ZLIBCHNG procedure prompts the operator for the new session library name and the menu name.



The specifications for the prompt screen and the contents of the ZLIBCHNG procedure are in *Listings for Sample Development Procedures*, which follows.

Listings for Sample Development Procedures

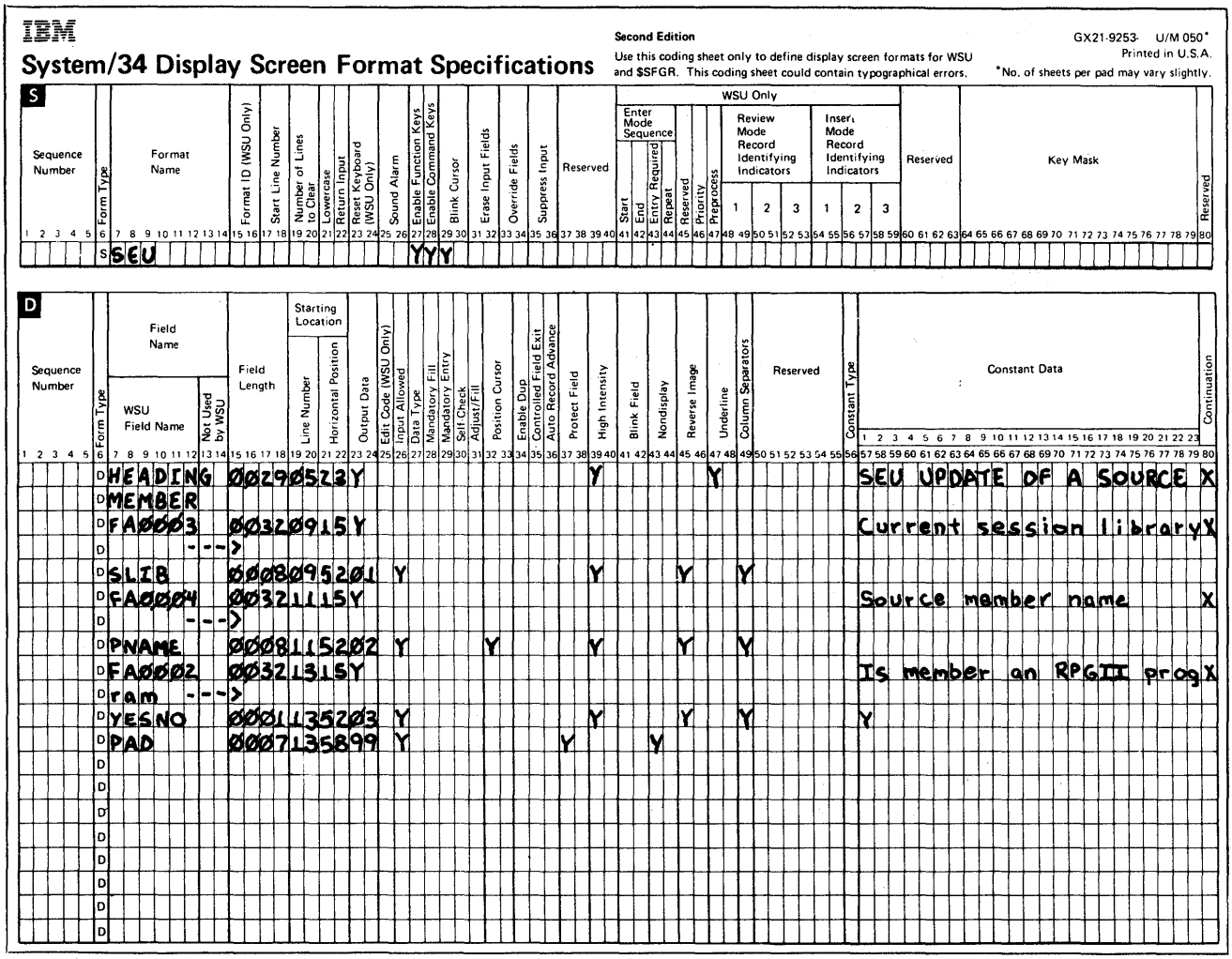
This section contains listings of the sample procedures just described. Along with the listing of procedure contents are the specifications for the prompt screens displayed by those procedures. The screens themselves are shown in the preceding sections.

Listings for ZSEUR

The ZSEUR procedure, which allows the user to update and recompile a program, contains the following statements:

```
* PROGRAM ENTRY FOR ZSEUR
// IFF ?1F'?SLIB?'/ PROMPT MEMBER-ZRFM,FORMAT-SEU
// IF ?3?/Y IF ?F3'R'
// IFF ?3?/Y IF ?F3'S'?/
SEU ?2R?,?3?,,,,?1?
// IF ?3?/S CANCEL
// PROMPT MEMBER-ZRFM, FORMAT-COMPILE
// IF ?3?/YES RPG ?2?,60,60,REPLACE,?1?,?1?,,,,?4??5?
// IFF ?3?/YES HELP RPG
```

As shown earlier, ZSEUR issues two displays. The first display (SEU) prompts for the session library and the source program name. The second display (COMPILE) prompts for information required to compile the program. The following charts are the display screen specifications for both displays.



IBM

System/34 Display Screen Format Specifications

Second Edition

Use this coding sheet only to define display screen formats for WSU and $SFGR. This coding sheet could contain typographical errors.

GX21-9253· U/M 050*
Printed in U.S.A.
*No. of sheets per pad may vary slightly.

S

Sequence Number	Form Type	Format Name	Format ID (WSU Only)	Start Line Number	Number of Lines to Clear	Lowercase	Return Input	Reset Keyboard (WSU Only)	Sound Alarm	Enable Function Keys	Enable Command Keys	Blink Cursor	Erase Input Fields	Override Fields	Suppress Input	Reserved	WSU Only: Enter Mode Sequence: Start	End	Entry Required	Repeat	Reserved	Priority	Preprocess	WSU Only: Review Mode Record Identifying Indicators: 1	2	3	WSU Only: Insert Mode Record Identifying Indicators: 1	2	3	Reserved	Key Mask	Reserved
1 2 3 4 5	6	7 8 9 10 11 12 13 14	15 16	17 18	19 20	21	22	23 24	25 26	27	28	29 30	31 32	33 34	35 36	37 38 39 40	41	42	43	44	45	46	47	48 49	50 51	52 53	54 55	56 57	58 59	60 61 62 63	64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	S	SEU								Y	Y																					

D

Sequence Number	Form Type	Field Name: WSU Field Name	Field Name: Not Used by WSU	Field Length	Starting Location: Line Number	Starting Location: Horizontal Position	Output Data	Edit Code (WSU Only)	Input Allowed	Data Type	Mandatory Fill	Mandatory Entry	Self Check	Adjust/Fill	Position Cursor	Enable Dup	Controlled Field Exit	Auto Record Advance	Protect Field	High Intensity	Blink Field	Nondisplay	Reverse Image	Underline	Column Separators	Reserved	Constant Type	Constant Data	Continuation
1 2 3 4 5	6	7 8 9 10 11 12	13 14	15 16 17 18	19 20	21 22	23 24	25	26	27	28	29	30	31	32 33	34	35	36	37 38	39 40	41 42	43 44	45 46	47 48	49	50 51 52 53 54 55	56	57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	D	HEADING		002905	2	3	Y													Y				Y				SEU UPDATE OF A SOURCE	X
	D	MEMBER																											
	D	FA0003		00320	9	15	Y																					Current session library	X
	D	-->																											
	D	SLIB		000809	5	201	Y													Y			Y		Y				
	D	FA0004		00321	1	15	Y																					Source member name	X
	D	-->																											
	D	PNAME		000811	5	202	Y							Y						Y			Y		Y				
	D	FA0002		00321	3	15	Y																					Is member an RPGIII prog	X
	D	ram -->																											
	D	YESNO		000111	35	203	Y													Y			Y		Y			Y	
	D	PAD		000711	35	899	Y												Y			Y							
	D																												
	D																												
	D																												
	D																												
	D																												
	D																												
	D																												
	D																												

IBM

System/34 Display Screen Format Specifications

Second Edition

Use this coding sheet only to define display screen formats for WSU and $SFGR. This coding sheet could contain typographical errors.

GX21-9253- U/M 050*
Printed in U.S.A.
*No. of sheets per pad may vary slightly.

S

Sequence Number	Form Type	Format Name	Format ID (WSU Only)	Start Line Number	Number of Lines to Clear	Lowercase	Return Input	Reset Keyboard (WSU Only)	Sound Alarm	Enable Function Keys	Enable Command Keys	Blink Cursor	Erase Input Fields	Override Fields	Suppress Input	Reserved	WSU Only: Enter Mode Sequence: Start	End	Entry Required	Repeat	Reserved	Priority	Preprocess	WSU Only: Review Mode Record Identifying Indicators: 1	2	3	WSU Only: Insert Mode Record Identifying Indicators: 1	2	3	Reserved	Key Mask	Reserved
1 2 3 4 5	6	7 8 9 10 11 12 13 14	15 16	17 18	19 20	21	22	23 24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48 49	50 51 52 53	54 55	56 57 58 59	60 61 62 63	64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	S	COMPILE										Y																					

D

Sequence Number	Form Type	Field Name: WSU Field Name	Field Name: Not Used by WSU	Field Length	Starting Location: Line Number	Starting Location: Horizontal Position	Output Data	Edit Code (WSU Only)	Input Allowed	Data Type	Mandatory Fill	Mandatory Entry	Self Check	Adjust/Fill	Position Cursor	Enable Dup	Controlled Field Exit	Auto Record Advance	Protect Field	High Intensity	Blink Field	Nondisplay	Reverse Image	Underline	Column Separators	Reserved	Constant Type	Constant Data	Continuation
1 2 3 4 5	6	7 8 9 10 11 12	13 14	15 16 17 18	19 20	21 22	23 24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49 50 51 52 53 54 55	56	57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	D	TITLE		0032	05	24	Y													Y				Y					COMPILE OPTION OF RPG IX	
	D	I PROGRAM																												
	D	FA0001		0032	09	16	Y																						Current session libraryX	
	D	-->																												
	D	SLIB		0008	09	52	01	Y												Y			Y		Y					
	D	FA0002		0032	11	16	Y																						Source program name X	
	D	-->																												
	D	PRGNAM		0006	11	52	02	Y												Y			Y		Y					
	D	PAD		0002	11	60	99	Y											Y											
	D	FA0003		0032	13	16	Y																						Standard default compilX	
	D	e -->																												
	D	YES		0003	13	52	03	Y							Y					Y			Y		Y				YES	
	D	PAD		0005	13	57	99	Y											Y											
	D	FA0004		0032	15	16	Y																						Run from JOBQ (Ø-no,1-X	
	D	yes) -->																												
	D	JOBQ		0000	11	55	20	4	YN											Y			Y		Y				1	
	D	PAD		0007	15	56	Y		Y										Y			Y								
	D	FA0005		0032	17	16	Y																						XREF list req (Ø-no,1-X	
	D	yes) -->																												
	D	XREF		0001	17	52	05	Y	N											Y				Y		Y			Ø	

IBM

System/34 Display Screen Format Specifications

Second Edition

Use this coding sheet only to define display screen formats for WSU and $SFGR. This coding sheet could contain typographical errors.

GX21-9253- U/M 050*
Printed in U.S.A.
*No. of sheets per pad may vary slightly.

S

Sequence Number (1–5); Form Type (6); Format Name (7–14); Format ID (WSU Only) (15–16); Start Line Number (17–18); Number of Lines to Clear (19–20); Lowercase (21); Return Input (22); Reset Keyboard (WSU Only) (23–24); Sound Alarm (25–26); Enable Function Keys (27); Enable Command Keys (28); Blink Cursor (29–30); Erase Input Fields (31–32); Override Fields (33–34); Suppress Input (35–36); Reserved (37–40); WSU Only: Enter Mode Sequence: Start (41), End (42), Entry Required (43), Repeat (44); Reserved (45); Priority (46); Preprocess (47); Review Mode Record Identifying Indicators 1 2 3 (48–53); Insert Mode Record Identifying Indicators 1 2 3 (54–59); Reserved (60–63); Key Mask (64–79); Reserved (80)

```
     S
```

D

Sequence Number (1–5); Form Type (6); Field Name: WSU Field Name (7–12), Not Used by WSU (13–14); Field Length (15–18); Starting Location: Line Number (19–20), Horizontal Position (21–22); Output Data (23–24); Edit Code (WSU Only) (25); Input Allowed (26); Data Type (27); Mandatory Fill (28); Mandatory Entry (29); Self-Check (30); Adjust/Fill (31); Position Cursor (32–33); Enable Dup (34); Controlled Field Exit (35); Auto Record Advance (36); Protect Field (37–38); High Intensity (39–40); Blink Field (41–42); Nondisplay (43–44); Reverse Image (45–46); Underline (47–48); Column Separators (49); Reserved (50–55); Constant Type (56); Constant Data 1–23 (57–79); Continuation (80)

```
     DPAD     0007175699 Y          Y     Y
     DFA0002  00291907Y                                 (Standard Default paramX
     Deters:
     DFA0007  00562007Y                                  Replace assumed, sourcX
     De and load processed from current
     DFA0008  00492108Y                                 session library.  60 BIX
     Docks used for work files.)
     D
     D
     D
     D
     D
     D
     D
     D
     D
     D
     D
     D
```

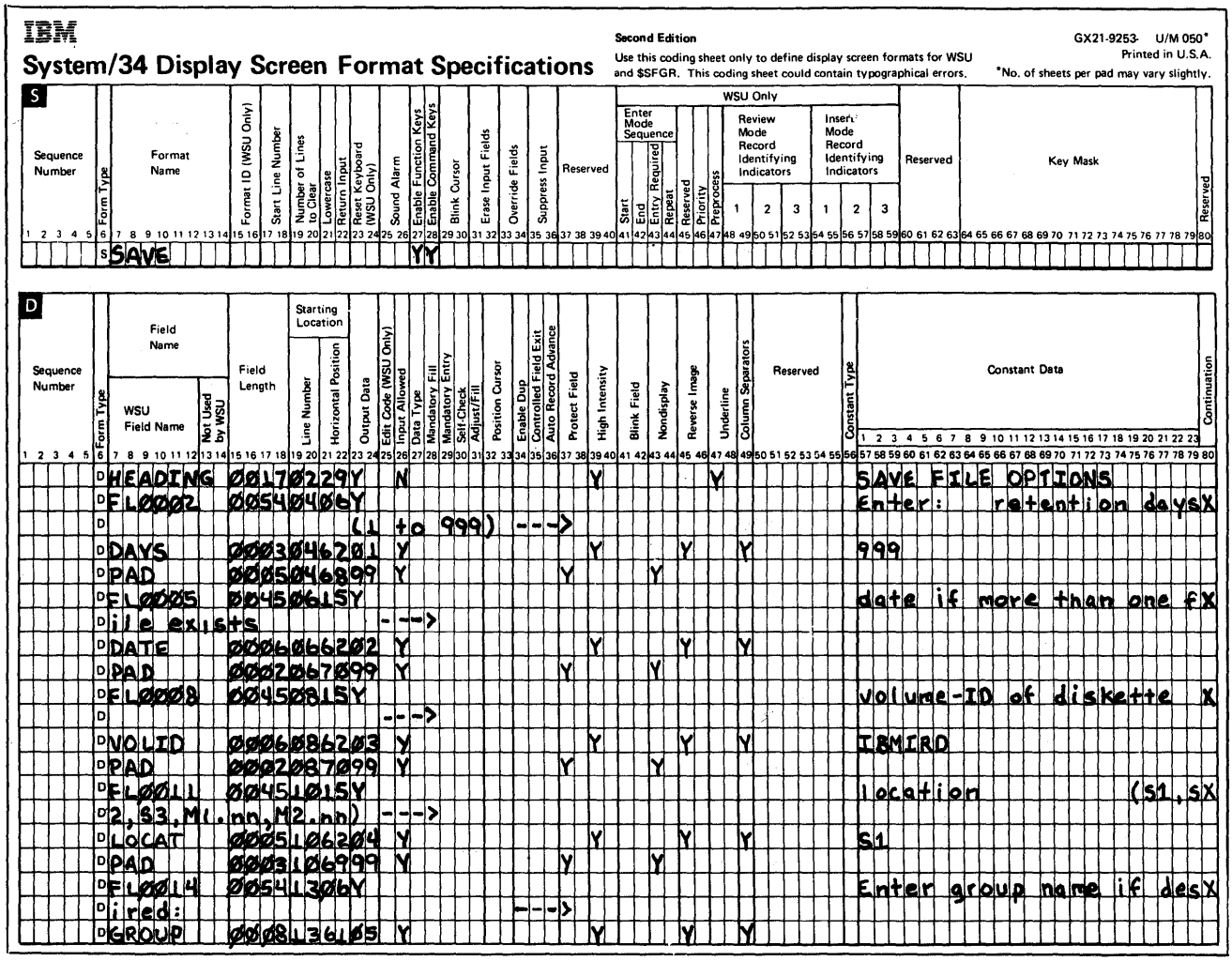
Listings for ZSAVEF

The ZSAVEF procedure, which allows the user to copy all files in a file group or to copy up to five individual files, contains the following statements:

```
* SAVE FILES
// PROMPT MEMBER-ZXFM,FORMAT-SAVE
// IF ?5?/ GOTO SKP1
// * '?5? GROUP FILES are BEING SAVED ON DISKETTE'
SAVE ALL,?1?,?2?,?3?,?5?,?4?
// TAG SKP1
// IF ?6?/ GOTO SKP2
// * '?6? FILE IS BEING SAVED ON DISKETTE'
//     LOAD $COPY
//     FILE NAME-COPYIN,LABEL-?6?,UNIT-F1
//     FILE NAME-COPYO,RETAIN-?1?,LABEL-?6?,LOCATION-?4?,AUTO-YES,
//     PACK-?3?,UNIT-I1
//     RUN
// COPYFILE OUTPUT-DISK,REORG-NO
// END
// TAG SKP2
// IF ?7? GOTO SKP3
// * '?7? FILE IS BEING SAVED ON DISKETTE'
//     LOAD $COPY
//     FILE NAME-COPYIN,LABEL-?7?,UNIT-F1
//     FILE NAME-COPYO,RETAIN-?1?,LABEL-?7?,LOCATION-?4?,AUTO-YES,
//     PACK-?3?,UNIT-I1
//     RUN
// COPYFILE OUTPUT-DISK,REORG-NO
// END
// TAG SKP3
// IF ?8?/ GOTO SKP4
// * '?8? FILE IS BEING SAVED ON DISKETTE'
//     LOAD $COPY
//     FILE NAME-COPYIN,LABEL-?8?,UNIT-F1
//     FILE NAME-COPYO,RETAIN-?1?,LABEL-?8?,LOCATION-?4?,AUTO-YES,
//     PACK-?3?,UNIT-I1
//     RUN
// COPYFILE OUTPUT-DISK,REORG-NO
// END
// TAG SKP4
// IF ?9?/ GOTO SKP5
// * '?9?/ FILE IS BEING SAVED ON DISKETTE'
//     LOAD $COPY
//     FILE NAME-COPYIN,LABEL-?9?,UNIT-F1
//     FILE NAME-COPYO,RETAIN-?1?,LABEL-?9?,LOCATION-?4?,AUTO-YES,
//     PACK-?3?,UNIT-I1
//     RUN
// COPYFILE OUPUT-DISK, REORG-NO
// END
```

```
// TAG SKP5
// IF ?10?/ GOTO SKP6
// * '?10? FILE IS BEING SAVED ON DISKETTE'
//     LOAD $COPY
//     FILE NAME-COPYIN,LABEL-?10?,UNIT-F1
//     FILE NAME-COPYO,RETAIN-?1?,LABEL-?10?,LABEL-?10?,LOCATION-?4?,AUTO-YES,
//     PACK-?3?,UNIT-I1
//     RUN
// COPYFILE OUTPUT-DISK,REORG-NO
// END
// TAG SKP6
```

As shown earlier, ZSAVEF issues one display (called SAVE), which prompts for the save file options. The following listing shows the display screen specifications for the display:



IBM

System/34 Display Screen Format Specifications

Second Edition

Use this coding sheet only to define display screen formats for WSU and $SFGR. This coding sheet could contain typographical errors.

GX21-9253- U/M 050*
Printed in U.S.A.
*No. of sheets per pad may vary slightly.

S

Sequence Number	Form Type	Format Name	Format ID (WSU Only)	Start Line Number	Number of Lines to Clear	Lowercase	Return Input	Reset Keyboard (WSU Only)	Sound Alarm	Enable Function Keys	Enable Command Keys	Blink Cursor	Erase Input Fields	Override Fields	Suppress Input	Reserved	WSU Only: Enter Mode Sequence: Start	End	Entry Required	Repeat	Reserved	Priority	Preprocess	WSU Only: Review Mode Record Identifying Indicators: 1	2	3	WSU Only: Insert Mode Record Identifying Indicators: 1	2	3	Reserved	Key Mask	Reserved
1 2 3 4 5	6	7 8 9 10 11 12 13 14	15 16	17 18	19 20	21	22	23 24	25	26	27 28	29 30	31 32	33 34	35 36	37 38 39 40	41	42	43	44	45	46	47	48 49	50 51	52 53	54 55	56 57	58 59	60 61 62 63	64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	S	SAVE									YY																					

D

Sequence Number	Form Type	Field Name: WSU Field Name	Field Name: Not Used by WSU	Field Length	Starting Location: Line Number	Starting Location: Horizontal Position	Output Data	Edit Code (WSU Only)	Input Allowed	Data Type	Mandatory Fill	Mandatory Entry	Self-Check	Adjust/Fill	Position Cursor	Enable Dup	Controlled Field Exit	Auto Record Advance	Protect Field	High Intensity	Blink Field	Nondisplay	Reverse Image	Underline	Column Separators	Reserved	Constant Type	Constant Data	Continuation
1 2 3 4 5	6	7 8 9 10 11 12	13 14	15 16 17 18	19 20	21 22	23 24	25	26	27	28	29	30	31	32 33	34	35	36	37	38	39 40	41	42 43	44	45	46	47 48	49 50 51 52 53 54 55	56	57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	D	HEADING		001702			29Y		N											Y				Y				SAVE FILE OPTIONS	
	D	FL0002		005404			06Y																					Enter: retention daysX	
	D						(1	to	999)	---	>																		
	D	DAYS		000304	62	01	Y													Y			Y		Y			999	
	D	PAD		000504	68	99	Y												Y			Y							
	D	FL0005		004506	15	Y																						date if more than one fX	
	D	ile exists						-	-->																				
	D	DATE		000606	62	02	Y													Y			Y		Y				
	D	PAD		000206	70	99	Y												Y			Y							
	D	FL0008		004508	15	Y																						volume-ID of diskette X	
	D								---	>																			
	D	VOLID		000608	62	03	Y													Y			Y		Y			IBMIRD	
	D	PAD		000208	70	99	Y												Y			Y							
	D	FL0011		004510	15	Y																						location (S1,SX	
	D	2,S3,M(,nn,M2,nn)						-	---	>																			
	D	LOCAT		000510	62	04	Y													Y			Y		Y			S1	
	D	PAD		000310	69	99	Y												Y			Y							
	D	FL0014		005413	06	Y																						Enter group name if desX	
	D	ired:														---	>												
	D	GROUP		000813	61	05	Y													Y			Y		Y				

IBM

System/34 Display Screen Format Specifications

Second Edition

Use this coding sheet only to define display screen formats for WSU and $SFGR. This coding sheet could contain typographical errors.

GX21-9253- U/M 050*
Printed in U.S.A.
*No. of sheets per pad may vary slightly.

S

Sequence Number	Form Type	Format Name	Format ID (WSU Only)	Start Line Number	Number of Lines to Clear	Lowercase	Return Input	Reset Keyboard (WSU Only)	Sound Alarm	Enable Function Keys	Enable Command Keys	Blink Cursor	Erase Input Fields	Override Fields	Suppress Input	Reserved	WSU Only: Enter Mode Sequence (Start, End)	WSU Only: Entry Required	WSU Only: Repeat	WSU Only: Reserved	WSU Only: Priority	WSU Only: Preprocess	WSU Only: Review Mode Record Identifying Indicators (1, 2, 3)	WSU Only: Insert Mode Record Identifying Indicators (1, 2, 3)	Reserved	Key Mask	Reserved
1 2 3 4 5	6	7 8 9 10 11 12 13 14	15 16	17 18	19 20	21	22	23 24	25 26	27	28	29 30	31 32	33 34	35 36	37 38 39 40	41 42	43	44	45	46	47	48 49 50 51 52 53	54 55 56 57 58 59	60 61 62 63	64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	S																										

D

Sequence Number	Form Type	Field Name: WSU Field Name	Field Name: Not Used by WSU	Field Length	Starting Location: Line Number	Starting Location: Horizontal Position	Output Data	Edit Code (WSU Only)	Input Allowed	Data Type	Mandatory Fill	Mandatory Entry	Self-Check	Adjust/Fill	Position Cursor	Enable Dup	Controlled Field Exit	Auto Record Advance	Protect Field	High Intensity	Blink Field	Nondisplay	Reverse Image	Underline	Column Separators	Reserved	Constant Type	Constant Data	Continuation
1 2 3 4 5	6	7 8 9 10 11 12	13 14	15 16 17 18	19 20	21 22	23 24	25	26	27	28	29 30	31	32	33 34	35	36	37	38	39 40	41 42	43 44	45 46	47 48	49	50 51 52 53 54 55	56	57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	D	FL0016		0054	16	06	Y									-											Enter up to 5 file nameX	
	D	$:														-	-	>										
	D	NAME1		0008	16	62	06	Y						Y					Y				Y		Y			
	D	FL0018		0051	18	09	Y																				- all files on diskettX	
	D								(ALL)				-	-	>													
	D	NAME2		0008	18	62	07	Y											Y				Y		Y			
	D	FL0020		0051	20	09	Y																				- an individual file	
	D	-							(name)				-	-	>													
	D	NAME3		0008	20	62	08	Y											Y				Y		Y			
	D	FL0022		0042	25	6	Y																				-->	
	D	NAME4		0008	22	62	09	Y											Y				Y		Y			
	D	FL0024		0042	45	6	Y																				-->	
	D	NAME5		0008	24	62	10	Y											Y				Y		Y			
	D																											
	D																											
	D																											
	D																											
	D																											
	D																											
	D																											

The ZLIBCHNG procedure, which allows the user to change the session library and/or the current menu, contains the following statements:

```
* PROCEDURE TO CHANGE SESSION LIBRARY (ZLIBCHNG)
// IFF ?1F'?SLIB?'?/ PROMPT MEMBER-ZRFM,FORMAT-LIB
// IFF ?2?/ LIBRARY NAME-?2?,SESSION-YES
// IFF ?2?/ IFF ?3?/ MENU?3?,?2?
// IF ?2?/ IFF ?3?/ MENU?3?
```

As shown earlier, ZLIBCHNG issues one display (called LIB), which prompts for the session library name and the menu name. The following listing shows the display screen specifications for the display:

IBM

System/34 Display Screen Format Specifications

Second Edition

Use this coding sheet only to define display screen formats for WSU and $SFGR. This coding sheet could contain typographical errors.

GX21-9253- U/M 050*
Printed in U.S.A.
*No. of sheets per pad may vary slightly.

S

Sequence Number	Form Type	Format Name	Format ID (WSU Only)	Start Line Number	Number of Lines to Clear	Lowercase	Return Input	Reset Keyboard (WSU Only)	Sound Alarm	Enable Function Keys	Enable Command Keys	Blink Cursor	Erase Input Fields	Override Fields	Suppress Input	Reserved	WSU Only: Enter Mode Sequence: Start	End	Entry Required	Repeat	Reserved	Priority	Preprocess	WSU Only: Review Mode Record Identifying Indicators: 1	2	3	WSU Only: Insert Mode Record Identifying Indicators: 1	2	3	Reserved	Key Mask	Reserved
1 2 3 4 5	6	7 8 9 10 11 12 13 14	15 16	17 18	19 20	21	22	23 24	25	26	27	28 29 30	31	32	33	34 35 36 37 38 39 40	41	42	43	44	45	46	47	48 49	50	51 52 53	54	55	56 57 58	59 60 61 62 63	64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	S	LIB										Y																				

D

Sequence Number	Form Type	Field Name: WSU Field Name	Field Name: Not Used by WSU	Field Length	Starting Location: Line Number	Starting Location: Horizontal Position	Output Data	Edit Code (WSU Only)	Input Allowed	Data Type	Mandatory Fill	Mandatory Entry	Self-Check	Adjust/Fill	Position Cursor	Enable Dup	Controlled Field Exit	Auto Record Advance	Protect Field	High Intensity	Blink Field	Nondisplay	Reverse Image	Underline	Column Separators	Reserved	Constant Type	Constant Data	Continuation
1 2 3 4 5	6	7 8 9 10 11 12	13 14	15 16 17 18	19 20	21 22	23 24	25	26	27	28	29	30	31	32 33	34	35	36	37 38	39 40	41 42	43 44	45 46	47 48	49	50 51 52 53 54 55	56	57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79	80
	D	FL0001		0022	04	27	Y													Y					Y			CHANGE SESSION LIBRARY	
	D	FL0002		0027	07	19	Y																					Current session libraryX	
	D	-->																											
	D	LIBRARY		0008	07	49	01	Y										Y	Y											
	D	FL0004		0028	10	19	Y																					New session library namX	
	D	~-->																											
	D	NEWLIB		0008	10	49	02	Y					Y							Y			Y		Y				
	D	FL0006		0020	13	19	Y																					Display new menu -->	
	D	MENU		0006	13	49	03	Y												Y			Y		Y				
	D	PAD		0002	13	59	99	Y											Y			Y							
	D	C0		0001	17	14	Y																					(	
	D	C1		0012	17	16	Y																					Enter either	
	D	C2		0012	17	29	Y													Y				Y				library name	
	D	C3		0002	17	42	Y																					or	
	D	C4		0009	17	45	Y													Y				Y				menu name	
	D	C5		0002	17	55	Y																					or	
	D	C6		0004	17	58	Y													Y				Y				both	
	D	C7		0001	17	63	Y																					)	
	D																												
	D																												

CREATING DISPLAY SCREEN FORMATS

To create the display screen formats used in this example, you could select item 1 from the TESTM menu. All display screen formats used by one program must be placed in a single display screen format load member. For RPG II programs, the member name must be the name of the program followed by the characters FM. If you select item 1 from the menu, you will be prompted for the name of the program that uses the formats. If you enter ORDHDR, the following command statement is generated:

```
SDA ORDHDRFM,TESTLIB,ORDHDRFM
```

When the SDA menu appears, you could select option 1 and then enter an N for the column indicator mode so that the first line on the screen can be used. Using the screen formats designed earlier, enter the information from the screen layout sheet onto the SDA blank screen. When the entire screen has been entered, you can press command key 9.

In this example, the developer would like to be prompted for field attributes; therefore, an * is entered in front of each input or output field, a *c* is entered in front of each constant, and a *t* is entered after each field.

An alternative method of prompting for attributes could be used by entering a Y for automatic prompting on the initial SDA menu. For this method, I for input and E or B for output are used rather than asterisks. SDA automatically prompts for additional attributes for these fields.

For screen E1 in display screen format ORDSE1, Figures 5-14 and 5-15 show the screen entry and attribute screens.

For screen E2 in display screen format ORDSE2, Figures 5-16 and 5-17 show the screen entry and attribute screens.

For screen E3 in display screen format ORDSE3, Figures 5-18, 5-19, and 5-20 show the S specification, screen entry, and attribute screens.

Figure 5-21 shows the $SFGR output generated when screens E1, E2, and E3 are built. All the display screen formats used by the ORDHDR program are built in one SDA run.

If changes are later required to formats that have been built, you can use the SDA update function to make changes. This function allows you to use SEU to make changes and automatically regenerates formats. An alternative method of changing formats is to use SEU directly and then use the FORMAT procedure to regenerate the formats.

Because the next program, ORDITM, uses screens E1 and E3, which were designed already, only screen E4 is created. Figures 5-22, 5-23, and 5-24 show the S specification, screen entry, and the attribute screens for screen E4. The specifications for screens E1 and E3 are included using SEU. Figure 5-25 shows the $SFGR output generated when the display screen formats used by ORDITM are built.

```
E1
                          O R D E R   E N T R Y

        CUSTOMER NUMBER     XXXXXX
        ORDER NUMBER        XXXXXX









        Press ENTER REC/ADV        - or -      CK1 - TO ENTER MISC ORDER INFO
                                               CK7 - CANCEL ORDER ENTRY
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM
```

Figure 5-14. Screen Entry for Screen E1

```
*E1t
                         *O R D E R   E N T R Yt

        cCUSTOMER NUMBERt   *XXXXXXt
        cORDER NUMBERt      *XXXXXXt









        cPress ENTER REC/ADV       - or -      CK1 - TO ENTER MISC ORDER INFOt
                                              cCK7 - CANCEL ORDER ENTRYt     *
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMt
```

Figure 5-15. Attribute Screen for Screen E1

```
E2
                          O R D E R  E N T R Y

   CUST NO   XXXXXX         ORDER NO   XXXXXX

SOLD TO XXXXXXXXXXXXXXXXXXXXX        SHIP TO XXXXXXXXXXXXXXXXXXXXXXXX
        XXXXXXXXXXXXXXXXXXXXX                XXXXXXXXXXXXXXXXXXXXXXXX
        XXXXXXXXXXXXXXXXXXX XX XXXXX         XXXXXXXXXXXXXXXXXXXXXXX XX XXXXX

CUST PO XXXXXXXXXX     SALESMAN NO XX
```

Figure 5-16. Screen Entry for Screen E2

```
*E2t
                          *O R D E R  E N T R Yt

   cCUST NOt *XXXXXXt       cORDER NOt *XXXXXXt

cSOLD TO*XXXXXXXXXXXXXXXXXXXXXXt      cSHIP TO*XXXXXXXXXXXXXXXXXXXXXXXXt
        *XXXXXXXXXXXXXXXXXXXXXXt              *XXXXXXXXXXXXXXXXXXXXXXXXt
        *XXXXXXXXXXXXXXXXXXX*XX*XXXXXt        *XXXXXXXXXXXXXXXXXXXXXXX*XX*XXXXXt

cCUST PO*XXXXXXXXXXt   cSALESMAN NO*XXt
```

Figure 5-17. Attribute Screen for Screen E2

```
FORMAT NAME                    ORDSE3
WSU FORMAT ID
START LINE NUMBER
NUMBER OF LINES TO CLEAR
LOWERCASE ALLOWED
RETURN INPUT
RESET KEYBOARD
SOUND ALARM
ENABLE FUNCTION KEYS                ENABLE COMMAND KEYS
BLINK CURSOR
ERASE INPUT FIELDS
OVERRIDE FIELDS                99
SUPPRES INPUT
KEY MASK

                              ****** WSU ONLY *******
                    ENTER MODE SEQUENCE
START-          END-          REQUIRED-          REPEAT-
PRIORITY-            PREPROCESS-
                    REVIEW MODE INDICATORS
RECORD ID 1-              RECORD ID 2-          RECORD ID 3-
                    INSERT MODE INDICATORS
INSERT ID 1-              INSERT ID 2-          INSERT ID 3-
```

Figure 5-18. S Specification Display for Screen E3

```
E3
                              O R D E R   E N T R Y

   CUST NO   XXXXXX           ORDER NO   XXXXXX

SOLD TO XXXXXXXXXXXXXXXXXXXXX          SHIP TO XXXXXXXXXXXXXXXXXXXXXXXXX
        XXXXXXXXXXXXXXXXXXXXX                  XXXXXXXXXXXXXXXXXXXXXXXXX
        XXXXXXXXXXXXXXXXXXX XX XXXXX           XXXXXXXXXXXXXXXXXXXXX XX XXXXX

CUST PO XXXXXXXXXX     SALESMAN NO XX

LINE  ITEM NO   QTY     DESCRIPTION              PRICE     AMOUNT




 01   XXXXXX   XXXXXX-  XXXXXXXXXXXXXXXXXXX   XXXXXXX    XXXXXXXX-
                                                CK2 - END OF ORDER
                                                CK3 - PAGE BACKWARDS ON ITEMS
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMM  CK8 - CANCEL THIS ORDER
```

Figure 5-19. Screen Entry for Screen E3

```
*E3t
                           *O R D E R  E N T R Y t

   cCUST NOt *XXXXXXt       cORDER NOt *XXXXXXt

 cSOLD TO*XXXXXXXXXXXXXXXXXXXXXXt       cSHIP TO*XXXXXXXXXXXXXXXXXXXXXXXXXt
         *XXXXXXXXXXXXXXXXXXXXXXt                *XXXXXXXXXXXXXXXXXXXXXXXXXt
         *XXXXXXXXXXXXXXXXXX*XX*XXXXXt           *XXXXXXXXXXXXXXXXXXXXXX*XX*XXXXXt

 cCUST POt*XXXXXXXXXXt   cSALESMAN NO*XXt

cLINE  ITEM NO   QTY     DESCRIPTION             PRICE     AMOUNTt




 *01t *XXXXXXt *XXXXXX-t *XXXXXXXXXXXXXXXXXXXXXt *XXXXXXXt  *XXXXXXXX-t
                                                 cCK2 - END OF ORDERt
                                                 cCK3 - PAGE BACKWARDS ON ITEMSt
MMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMMtcCK8 - CANCEL THIS ORDERt
```

Figure 5-20. Attribute Screen for Screen E3

```
  SOURCE INPUT SCREEN FORMAT SOURCE SPECIFICATIONS                    DATE  02/12/79     TIME  08.34

00010SORDSE1                        99
00020DSCODE   00020103Y  Y           Y     Y           CE1
00030DTITLE   00210231Y                        Y      CO R D E R   E N T R Y
00040DFL0003  00150511Y                               CCUSTOMER NUMBER
00050DCUSNO   00060531Y  Y  Y ZY      Y         Y
00060DFL0005  00120711Y                               CORDER NUMBER
00070DORDNO   00060731Y  Y  Y B       Y         Y
00080DFL0007  00702011Y                               CP      ENTER REC/ADV    X
00090D   -    -         CK1 - TO ENTER MISC ORDER INFO
00100DFL0008  00242251Y                               CCK7 - CANCEL ORDER ENTRX
00110DY
00120DERRMSG  0048230199                    99


  EXECUTION TIME OUTPUT BUFFER DESCRIPTION

  FIELD                          START          END
  NAME            LENGTH         POSITION       POSITION

  CUSNO              6              1              6
  ORDNO              6              7             12
  ERRMSG            48             13             60


  INPUT BUFFER DESCRIPTION

  FIELD                          START          END
  NAME            LENGTH         POSITION       POSITION

  SCODE              2              1              2
  CUSNO              6              3              8
  ORDNO              6              9             14


  SOURCE INPUT SCREEN FORMAT SOURCE SPECIFICATIONS

00130SORDSE2
00140DSCODE   00020103Y  Y           Y     Y           CF2
00150DTITLE   00210231Y                        Y      CO R D E R   E N T R Y
00160DFL0003  00070406Y                               CCUST NO
00170DCUSNO   00060416Y  Y           Y Y
00180DFL0005  00080431Y                               CORDER NO
00190DDRONO   00060442Y  Y           Y Y
00200DFL0007  00070603Y                               CSOLD TO
00210DCNAME   00250611Y  Y           Y Y
00220DFL0009  00070642Y                               CSHIP TO
00230DSNAME   00250650Y  Y     Y       Y
00240DCADDR   00250711Y  Y           Y Y
00250DSADDR   00250750Y  Y             Y
00260DCCITY   00220811Y  Y           Y Y
00270DCSTATE  00020834Y  Y           Y Y
00280DCZIPCD  00050837Y  Y           Y Y
00290DSCITY   00220850Y  Y             Y
00300DSSTATE  00020873Y  Y             Y
00310DSZIPCD  00050876Y  Y             Y
00320DFL0019  00071003Y                               CCUST PO
00330DCPONO   00101011Y  Y             Y
00340DFL0021  00111026Y                               CSALESMAN NO
00350DCSLSNO  00021038Y  Y             Y


  EXECUTION TIME OUTPUT BUFFER DESCRIPTION

  FIELD                          START          END
  NAME            LENGTH         POSITION       POSITION

  SCODE              2              1              2
  CUSNO              6              3              8
  ORDNO              6              9             14
  CNAME             25             15             39
  SNAME             25             40             64
  CADDR             25             65             89
  SADDR             25             90            114
  CCITY             22            115            136
  CSTATE             2            137            138
  CZIPCD             5            139            143
  SCITY             22            144            165
  SSTATE             2            166            167
  SZIPCD             5            168            172
  CPONO             10            173            182
  CSLSNO             2            183            184


  INPUT BUFFER DESCRIPTION

  FIELD                          START          END
  NAME            LENGTH         POSITION       POSITION

  SCODE              2              1              2
  CUSNO              6              3              8
  ORDNO              6              9             14
  CNAME             25             15             39
  SNAME             25             40             64
  CADDR             25             65             89
  SADDR             25             90            114
  CCITY             22            115            136
  CSTATE             2            137            138
  CZIPCD             5            139            143
  SCITY             22            144            165
  SSTATE             2            166            167
  SZIPCD             5            168            172
  CPONO             10            173            182
  CSLSNO             2            183            184
```

Figure 5-21 (Part 1 of 2). $SFGR Output for Display Screen Formats Used by ORDHDR

```
  SOURCE INPUT SCREEN FORMAT SOURCE SPECIFICATIONS                    DATE  02/12/79     TIME  08.34

00360DSORDSE3                      99
00370DSCODE    00020103Y  Y          Y     Y              CE3
00380DTITLE    00210231Y                      Y          CO R D E R   E N T R Y
00390DFL0003   00070406Y                                 CCUST NO
00400DCUSNO    00060416Y  Y          Y Y
00410DFL0005   00080431Y                                 CORDER NO
00420DORDNO    00060442Y  Y          Y Y
00430DFL0007   00070603Y                                 CSOLD TO
00440DCNAME    00250611Y  Y          Y Y
00450DFL0009   00070642Y                                 CSHIP TO
00460DSNAME    00250650Y  Y            Y
00470DCADDR    00250711Y  Y          Y Y
00480DSADDR    00250750Y  Y            Y
00490DCCITY    00220811Y  Y          Y Y
00500DCSTATE   00020834Y  Y          Y Y
00510DCZIPCD   00050837Y  Y          Y Y
00520DSCITY    00220850Y  Y            Y
00530DSSTATE   00020873Y  Y            Y
00540DSZIPCD   00050876Y  Y            Y
00550DFL0019   00071003Y                                 CCUST PO
00560DCPONO    00101011Y  Y            Y
00570DFL0021   00111026Y                                 CSALESMAN NO
00580DCSLSNO   00021038Y  Y            Y
00590DFL0001   00651203Y                                 CLINE  ITEM NO   QTY    X
00600D DESCRIPTION             PRICE      AMOUNT
00610DELINE    00022003Y  Y            Y
00620DITEMNO   00062008Y  Y    BY      Y
00630DQTY      00062017Y  Y            Y
00640DDESCR    00202026Y  Y            Y
00650DPRICE    00062049Y  Y            Y
00660DAMOUNT   00082059Y  Y            Y
00670DFL0008   00182150Y                                 CCK2 - END OF ORDER
00680DFL0009   00292250Y                                 CCK3 - PAGE BACKWARDS ONX
00690D ITEMS
00700DERRMSG   0048230199                    99
00710DFL0011   00232350Y                                 CCK8 - CANCEL THIS ORDER


  EXECUTION TIME OUTPUT BUFFER DESCRIPTION

  FIELD                              START          END
  NAME              LENGTH           POSITION       POSITION

  CUSNO                6               1              6
  ORDNO                6               7             12
  CNAME               25              13             37
  SNAME               25              38             62
  CADDR               25              63             87
  SADDR               25              88            112
  CCITY               22             113            134
  CSTATE               2             135            136
  CZIPCD               5             137            141
  SCITY               22             142            163
  SSTATE               2             164            165
  SZIPCD               5             166            170
  CPONO               10             171            180
  CSLSNO               2             181            182
  ELINE                2             183            184
  ITEMNO               6             185            190
  QTY                  6             191            196
  DESCR               20             197            216
  PRICE                6             217            222
  AMOUNT               8             223            230
  ERRMSG              48             231            278


  INPUT BUFFER DESCRIPTION

  FIELD                              START          END
  NAME              LENGTH           POSITION       POSITION

  SCODE                2               1              2
  CUSNO                6               3              8
  ORDNO                6               9             14
  CNAME               25              15             39
  SNAME               25              40             64
  CADDR               25              65             89
  SADDR               25              90            114
  CCITY               22             115            136
  CSTATE               2             137            138
  CZIPCD               5             139            143
  SCITY               22             144            165
  SSTATE               2             166            167
  SZIPCD               5             168            172
  CPONO               10             173            182
  CSLSNO               2             183            184
  ELINE                2             185            186
  ITEMNO               6             187            192
  QTY                  6             193            198
  DESCR               20             199            218
  PRICE                6             219            224
  AMOUNT               8             225            232


  ORDHDRFM   SCREEN FORMAT LOAD MEMBER


FORMAT ORDSE1   REQUIRES   512 BYTES OF STORAGE
FORMAT ORDSE2   REQUIRES   768 BYTES OF STORAGE
FORMAT ORDSE3   REQUIRES  1024 BYTES OF STORAGE
```

Figure 5-21 (Part 2 of 2). $SFGR Output for Display Screen Formats Used by ORDHDR

```
FORMAT NAME                 ORDSE4
WSU FORMAT ID
START LINE NUMBER           V
NUMBER OF LINES TO CLEAR
LOWERCASE ALLOWED
RETURN INPUT
RESET KEYBOARD
SOUND ALARM
ENABLE FUNCTION KEYS              ENABLE COMMAND KEYS
BLINK CURSOR
ERASE INPUT FIELDS
OVERRIDE FIELDS
SUPPRES INPUT
KEY MASK

                           ****** WSU ONLY *******
                    ENTER MODE SEQUENCE
START-          END-        REQUIRED-        REPEAT-
PRIORITY-            PREPROCESS-
                    REVIEW MODE INDICATORS
RECORD ID 1-             RECORD ID 2-        RECORD ID 3-
                    INSERT MODE INDICATORS
INSERT ID 1-            INSERT ID 2-         INSERT ID 3-
```

Figure 5-22. S Specification Display for Display Screen E4

```
  XX  XXXXXX   XXXXXX-  XXXXXXXXXXXXXXXXXXXX XX,XXX.XX  XXX,XXX.XX-
```

Figure 5-23. Screen Entry for Display Screen E4

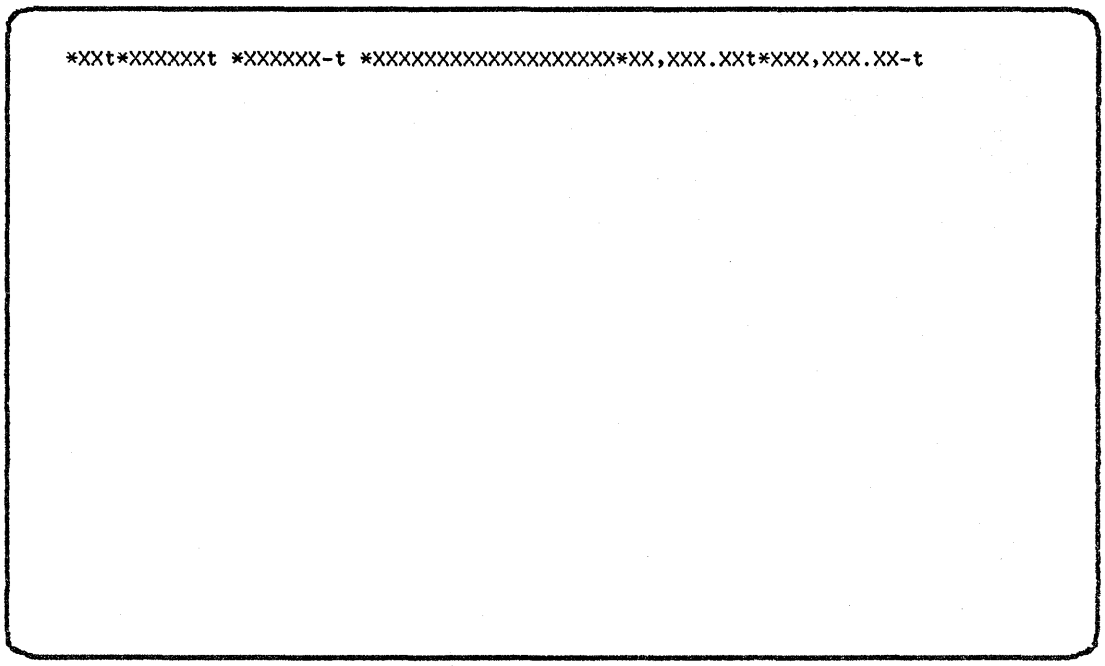


Figure 5-24. Attribute Screen for Display Screen E4

```
  SOURCE INPUT SCREEN FORMAT SOURCE SPECIFICATIONS                    DATE  02/12/79     TIME  08.33

00010SORDSE1                    99
00020DSCODE    00020103Y  Y           Y     Y           CE1
00030DTITLE    00210231Y                        Y       CO R D E R   E N T R Y
00040DFL0003   00150511Y                                CCUSTOMER NUMBER
00050DCUSNO    00060531Y  Y  Y ZY      Y         Y
00060DFL0005   00120711Y                                CORDER NUMBER
00070DORDNO    00060731Y  Y  Y B       Y         Y
00080DFL0007   00702011Y                                CP     ENTER REC/ADV    X
00090D   -     -        CK1 - TO ENTER MISC ORDER INFO
00100DFL0008   00242251Y                                CCK7 - CANCEL ORDER ENTRX
00110DY
00120DERRMSG   0048230199                  99
0013 D*


  EXECUTION TIME OUTPUT BUFFER DESCRIPTION

  FIELD                          START          END
  NAME             LENGTH        POSITION       POSITION

  CUSNO               6             1              6
  ORDNO               6             7             12
  ERRMSG             48            13             60


  INPUT BUFFER DESCRIPTION

  FIELD                          START          END
  NAME             LENGTH        POSITION       POSITION

  SCODE               2             1              2
  CUSNO               6             3              8
  ORDNO               6             9             14


  SOURCE INPUT SCREEN FORMAT SOURCE SPECIFICATIONS

00140SORDSE3                    99
00150DSCODE    00020103Y  Y           Y     Y           CE3
00160DTITLE    00210231Y                        Y       CO R D E R   E N T R Y
00170DFL0003   00070406Y                                CCUST NO
00180DCUSNO    00060416Y  Y           Y Y
00190DFL0005   00080431Y                                CORDER NO
00200DORDNO    00060442Y  Y           Y Y
00210DFL0007   00070603Y                                CSOLD TO
00220DCNAME    00250611Y  Y           Y Y
00230DFL0009   00070642Y                                CSHIP TO
00240DSNAME    00250650Y  Y             Y
00250DCADDR    00250711Y  Y           Y Y
00260DSADDR    00250750Y  Y             Y
00270DCCITY    00220811Y  Y           Y Y
00280DCSTATE   00020834Y  Y           Y Y
00290DCZIPCD   00050837Y  Y           Y Y
00300DSCITY    00220850Y  Y             Y
00310DSSTATE   00020873Y  Y             Y
00320DSZIPCD   00050876Y  Y             Y
00330DFL0019   00071003Y                                CCUST PO
00340DCPONO    00101011Y  Y             Y
00350DFL0021   00111026Y                                CSALESMAN NO
00360DCSLSNO   00021038Y  Y             Y
0037 DFL0001   00651203Y                                CLINE  ITEM NO   QTY    X
0038 D DESCRIPTION             PRICE      AMOUNT
0039 DELINE    00022003Y  Y             Y
0040 DITEMNO   00062008Y  Y   BY        Y
0041 DQTY      00062017Y  Y             Y
0042 DDESCR    00202026Y  Y             Y
0043 DPRICE    00062049Y  Y             Y
0044 DAMOUNT   00082059Y  Y             Y
0045 DFL0008   00182150Y                                CCK2 - END OF ORDER
0046 DFL0009   00292250Y                                CCK3 - PAGE BACKWARDS ONX
0047 D ITEMS
0048 DERRMSG   0048230199                  99
0049 DFL0011   00232350Y                                CCK8 - CANCEL THIS ORDER
0050 D*


  EXECUTION TIME OUTPUT BUFFER DESCRIPTION

  FIELD                          START          END
  NAME             LENGTH        POSITION       POSITION

  CUSNO               6             1              6
  ORDNO               6             7             12
  CNAME              25            13             37
  SNAME              25            38             62
  CADDR              25            63             87
  SADDR              25            88            112
  CCITY              22           113            134
  CSTATE              2           135            136
  CZIPCD              5           137            141
  SCITY              22           142            163
  SSTATE              2           164            165
  SZIPCD              5           166            170
  CPONO              10           171            180
  CSLSNO              2           181            182
  ELINE               2           183            184
  ITEMNO              6           185            190
  QTY                 6           191            196
  DESCR              20           197            216
  PRICE               6           217            222
  AMOUNT              8           223            230
  ERRMSG             48           231            278
```

Figure 5-25 (Part 1 of 2). $SFGR Output for Display Screen Formats Used by ORDITM

```
INPUT BUFFER DESCRIPTION

FIELD                                START          END
NAME              LENGTH             POSITION       POSITION

SCODE                2                  1              2
CUSNO                6                  3              8
ORDNO                6                  9             14
CNAME               25                 15             39
SNAME               25                 40             64
CADDR               25                 65             89
SADDR               25                 90            114
CCITY               22                115            136
CSTATE               2                137            138
CZIPCD               5                139            143
SCITY               22                144            165
SSTATE               2                166            167
SZIPCD               5                168            172
CPONO               10                173            182
CSLSNO               2                183            184
ELINE                2                185            186
ITEMNO               6                187            192
QTY                  6                193            198
DESCR               20                199            218
PRICE                6                219            224
AMOUNT               8                225            232


SOURCE INPUT SCREEN FORMAT SOURCE SPECIFICATIONS

00510SORDSE4    V
00520DLINENO  00020103Y                Y
00530DCUSNO   00060107Y                Y
00540DQTY     00070116Y                Y
00550DDESCR   00200126Y                Y
00560DPRICE   00090147Y                Y
00570DAMT     00110158Y                Y


EXECUTION TIME OUTPUT BUFFER DESCRIPTION

FIELD                                START          END
NAME              LENGTH             POSITION       POSITION

LINENO               2                  1              2
CUSNO                6                  3              8
QTY                  7                  9             15
DESCR               20                 16             35
PRICE                9                 36             44
AMT                 11                 45             55


ORDITMFM   SCREEN FORMAT LOAD MEMBER


FORMAT ORDSE1   REQUIRES   512 BYTES OF STORAGE
FORMAT ORDSE3   REQUIRES  1024 BYTES OF STORAGE
FORMAT ORDSE4   REQUIRES   256 BYTES OF STORAGE
```

Figure 5-25 (Part 2 of 2). $SFGR Output for Display Screen Formats Used by ORDITM

CODING THE PROGRAMS

Coding with RPG II

After the display screen formats are generated, you can use SDA to generate the RPG II specifications for the formats. To do this, select item 1 from the TESTM menu and enter the program name (for example, ORDHDR). Select option 8 on the SDA menu. Enter RPG for the type of program and the control specifications. Figure 5-26 shows the SDA entries required to build the ORDHDR program. After SDA generates the program, use item 20 from the TESTM menu to list the program. Figure 5-27 shows the ORDHDR RPG II program generated by SDA. You can follow the same procedures for the ORDITM program. Figure 5-28 shows the ORDITM RPG II program generated by SDA.

The file specifications are entered next. By executing item 4 from the TESTM menu, all the file specifications are entered under the name FILES. Figure 5-29 shows a listing of each of the file specifications. SEU will be used to copy the file specifications needed into each RPG II program.

Modular programming is used. That is, the program is modified to set on the proper indicators in the input and output specifications, and subroutines are also set up. The detail calculations will be added later. Using the program flowchart, basic program functions are easily identified.

It is easier to test and debug a smaller program than a larger one. If possible, do not code all functions into the program without testing parts of it. Use comments in the program to make testing and maintenance easier.

Figure 5-30 shows the ORDHDR RPG II program partially coded.

```
                SDA MENU


ENTER THE NUMBER ASSOCIATED WITH THE OPERATION YOU WOULD
LIKE TO PEFORM:

1 CREATE A NEW $SFGR/WSU SOURCE MEMBER
2 ADD TO AN EXISTING $SFGR/WSU SOURCE MEMBER
3 UPDATE AN EXISTING $SFGR/WSU SOURCE MEMBER
4 DISPLAY THE FORMAT IN AN EXISTING $SFGR OBJECT MEMBER
5 DELETE A FORMAT FROM AN EXISTING $SFGR/WSU SOURCE MEMBER
6 UPDATE EXISTING $SFGR/WSU SOURCE STATEMENTS VIA SEU
7 BUILD A MENU INTERACTIVELY
8 BUILD WSU PROGRAM OR RPG II SPECIFICATIONS FOR WORKSTN FILE

                                                          8

COL IND MODE?  ENTER Y OR N. DEFAULT IS Y.....................   Y
WSU FORMAT MEMBER? ENTER Y OR N. DEFAULT IS N.................   N
AUTOMATIC PROMPTING? ENTER Y OR N. DEFAULT IS N...............   N

          COMMAND FUNCTION KEY 7 TO END JOB
```

Figure 5-26 (Part 1 of 3). SDA Entries for Generating ORDHDR RPG II Program

```
ENTER RPG OR WSU ..... RPG
```

Figure 5-26 (Part 2 of 3). SDA Entries for Generating ORDHDR RPG II Program

```
                CONTROL SPECIFICATION ENTRIES

NUMBER OF FORMATS..................................... 05
NAME TO CALL RPG SOURCE MEMBER........................ ORDHDR

              WORKSTN FILE DESCRIPTION ENTRIES

NAME OF WORKSTN FILE.................................. SCREEN
RECORD LENGTH FROM $SFGR OUTPUT....................... 1000
NUMBER OF DISPLAY STATIONS.....................
NUMBER OF INDICATORS TO SAVE...................
NAME OF DATA STRUCTURE TO SAVE.................
NAME OF FIELD CONTAINING VARIABLE START LINE NUMBER .....
NAME OF FIELD CONTAINING DISPLAY STATION ID.....
NAME OF FORMAT LOAD MEMBER.....................
```

Figure 5-26 (Part 3 of 3). SDA Entries for Generating ORDHDR Program Header (H) Specification

```
  TESTLIB  MEMBER            DATE  03/09/79     TIME  13.27

TYPE  NAME        DISK ADDR       TOTAL  NUM TEXT/RECORD  ATTRIBUTES  LINK ADDR/NUM STMT  RLD DISP  ENTRY ADDR  PROG SIZE  MRT LEVEL

S  ORDHDR     377617/D5E311      7/0007      80/50        000000          88/0058                                              3

     H                                          05                      ORDHDR
     FSCREEN  CP  F *  1000           WORKSTN
     ISCREEN
     I* FORMAT- ORDSE1
     I                                    00010002 SCODE
     I                                    00030008 CUSNO
     I                                    00090014 ORDNO
     I* FORMAT- ORDSE2
     I                                    00010002 SCODE
     I                                    00030008 CUSNO
     I                                    00090014 ORDNO
     I                                    00150039 CNAME
     I                                    00400064 SNAME
     I                                    00650089 CADDR
     I                                    00900114 SADDR
     I                                    01150136 CCITY
     I                                    01370138 CSTATE
     I                                    01390143 CZIPCD
     I                                    01440165 SCITY
     I                                    01660167 SSTATE
     I                                    01680172 SZIPCD
     I                                    01730182 CPONO
     I                                    01830184 CSLSNO
     I* FORMAT- ORDSE3
     I                                    00010002 SCODE
     I                                    00030008 CUSNO
     I                                    00090014 ORDNO
     I                                    00150039 CNAME
     I                                    00400064 SNAME
     I                                    00650089 CADDR
     I                                    00900114 SADDR
     I                                    01150136 CCITY
     I                                    01370138 CSTATE
     I                                    01390143 CZIPCD
     I                                    01440165 SCITY
     I                                    01660167 SSTATE
     I                                    01680172 SZIPCD
     I                                    01730182 CPONO
     I                                    01830184 CSLSNO
     I                                    01850186 ELINE
     I                                    01870192 ITEMNO
     I                                    01930198 QTY
     I                                    01990218 DESCR
     I                                    02190224 PRICE
     I                                    02250232 AMOUNT
     OSCREEN
     O                                      K8 *ORDSE1  *
     O                          CUSNO    0006
     O                          ORDNO    0012
     O                99        ERRMSG   0060
     O                                      K8 *ORDSE2  *
     O                          SCODE    0002
     O                          CUSNO    0008
     O                          ORDNO    0014
     O                          CNAME    0039
     O                          SNAME    0064
     O                          CADDR    0089
     O                          SADDR    0114
     O                          CCITY    0136
     O                          CSTATE   0138
     O                          CZIPCD   0143
     O                          SCITY    0165
     O                          SSTATE   0167
     O                          SZIPCD   0172
     O                          CPONO    0182
     O                          CSLSNO   0184
     O                                      K8 *ORDSE3  *
     O                          CUSNO    0006
     O                          ORDNO    0012
     O                          CNAME    0037
     O                          SNAME    0062
     O                          CADDR    0087
     O                          SADDR    0112
     O                          CCITY    0134
     O                          CSTATE   0136
     O                          CZIPCD   0141
     O                          SCITY    0163
     O                          SSTATE   0165
     O                          SZIPCD   0170
     O                          CPONO    0180
     O                          CSLSNO   0182
     O                          ELINE    0184
     O                          ITEMNO   0190
     O                          QTY      0196
     O                          DESCR    0216
     O                          PRICE    0222
     O                          AMOUNT   0230
     O                99        ERRMSG   0278
```

Figure 5-27 Listing of ORDHDR RPG II Program

```
  TESTLIB  MEMBER            DATE  02/12/79     TIME  08.38

TYPE  NAME        DISK ADDR       TOTAL  NUM TEXT/RECORD  ATTRIBUTES  LINK ADDR/NUM STMT  RLD DISP  ENTRY ADDR  PROG SIZE  MRT LEVEL

 S  ORDITM      66328/010318       5/0005       80/50         000000          65/0041                                                  3

     H                                                  05                    ORDITM
     FSCREEN  CP  F    1000            WORKSTN
     F                                                   KSLN   SLINE
     ISCREEN
     I* FORMAT- ORDSE1
     I                                        00010002 SCODE
     I                                        00030008 CUSNO
     I                                        00090014 ORDNO
     I* FORMAT- ORDSE3
     I                                        00010002 SCODE
     I                                        00030008 CUSNO
     I                                        00090014 ORDNO
     I                                        00150039 CNAME
     I                                        00400064 SNAME
     I                                        00650089 CADDR
     I                                        00900114 SADDR
     I                                        01150136 CCITY
     I                                        01370138 CSTATE
     I                                        01390143 CZIPCD
     I                                        01440165 SCITY
     I                                        01660167 SSTATE
     I                                        01680172 SZIPCD
     I                                        01730182 CPONO
     I                                        01830184 CSLSNO
     I                                        01850186 ELINE
     I                                        01870192 ITEMNO
     I                                        01930198 QTY
     I                                        01990218 DESCR
     I                                        02190224 PRICE
     I                                        02250232 AMOUNT
     I* FORMAT- ORDSE4
     OSCREEN
     O                                  KB  *ORDSE1  *
     O                          CUSNO   0006
     O                          ORDNO   0012
     O                  99      ERRMSG  0060
     O                                  KB  *ORDSE3  *
     O                          CUSNO   0006
     O                          ORDNO   0012
     O                          CNAME   0037
     O                          SNAME   0062
     O                          CADDR   0087
     O                          SADDR   0112
     O                          CCITY   0134
     O                          CSTATE  0136
     O                          CZIPCD  0141
     O                          SCITY   0163
     O                          SSTATE  0165
     O                          SZIPCD  0170
     O                          CPONO   0180
     O                          CSLSNO  0182
     O                          ELINE   0184
     O                          ITEMNO  0190
     O                          QTY     0196
     O                          DESCR   0216
     O                          PRICE   0222
     O                          AMOUNT  0230
     O                  99      ERRMSG  0278
     O                                  KB  *ORDSE4  *
     O                          LINENO  0002
     O                          CUSNO   0009
     O                          QTY     0015
     O                          DESCR   0035
     O                          PRICE   0044
     O                          AMT     0055
```

Figure 5-28. Listing of ORDITM RPG II Program

```
  TESTLIB  MEMBER            DATE  12/29/78     TIME  17.57

TYPE  NAME        DISK ADDR        TOTAL  NUM TEXT/RECORD  ATTRIBUTES  LINK ADDR/NUM STMT  RLD DISP  ENTRY ADDR  PROG SIZE  MRT LEVEL

 S  FILES       221874/0362B2        7/0007       96/60         000000          71/0047                                                3

0001 FCMAST   IC  F 128  128R 6AI      4 DISK
0002 FSMAST   IC  F 128  128R 6AI      4 DISK
0003 FIMAST   UC  F 128  128R 6AI      4 DISK
0004 FTRANS   UC  F 128  128R          DISK
0005 FTRANSLOGUC  F 128  128R          DISK
0006 ICMAST   MA  11    1 CM   2 CA   3 C
0007 I                                          1   2 CRECCD
0008 I                                          3   3 CDELET
0009 I                                          4   90CUSNO
0010 I                                         10  34 CNAME
0011 I                                         35  59 CADDR
0012 I                                         60  81 CCITY
0013 I                                         82  83 CSTATE
0014 I                                         84  88OCZIPCD
0015 I                                         89  900CSLSNO
0016 ISMAST   SA  12    1 CS   2 CA   3 C
0017 I                                          1   2 SRECCD
0018 I                                          3   3 SDELET
0019 I                                          4   90SCUSNO
0020 I                                         10  34 SNAME
0021 I                                         35  59 SADDR
0022 I                                         60  81 SCITY
0023 I                                         82  83 SSTATE
0024 I                                         84  88OSZIPCD
0025 IIMAST   IT  13    1 CI   2 CT   3 C
0026 I                                          1   2 IRECCD
0027 I                                          3   3 IDELET
0028 I                                          4   90ITMNO
0029 I                                         10  29 IDESCR
0030 I                                         30  332IPRICE
0031 I                                         34  38 IWHLOC
0032 ITRANS   QU  01
0033 I                                          1   2 ORECCD
0034 I                                          3   50RR#1
0035 I                                          6   80RR#2
0036 I                                          9  110RR#3
0037 I        CU  15    1 CC   2 CU   3 C
0038 I                                          1   2 ORECCD
0039 I                                          3   3 ODELET
0040 I                                          4   90CUSNO
0041 I                                         10  150ORDNO
0042 I                                         16  40 CNAME
0043 I                                         41  65 CADDR
0044 I                                         66  87 CCITY
0045 I                                         88  89 CSTATE
0046 I                                         90  940CZIPCD
0047 I                                         95  960CSLSNO
0048 I                                         97 106 CPONO
0049 I        CS  16    1 CC   2 CS   3 C
0050 I                                          1   2 RCODE
0051 I                                          3   3 ODELET
0052 I                                          4   90CUSNO
0053 I                                         10  150ORDNO
0054 I                                         16  40 SNAME
0055 I                                         41  65 SADDR
0056 I                                         66  87 SCITY
0057 I                                         88  89 SSTATE
0058 I                                         90  940SZIPCD
0059 I        IT  17    1 CI   2 CT   3 C
0060 I                                          1   2 ORECCD
0061 I                                          3   3 ODELET
0062 I                                          4   90CUSNO
0063 I                                         10  150ORDNO
0064 I                                         16  170OLINE
0065 I                                         18  230ITMNO
0066 I                                         24  43 IDESCR
0067 I                                         44  490OQTY
0068 I                                         50  552OPRICE
0069 I                                         56  632OAMT
0070 I                                         64  68 IWHLOC
0071 I            18
```

Figure 5-29. Listing of File Specifications

```
  TESTLIB  MEMBER            DATE  12/29/78     TIME  17.57

TYPE  NAME         DISK ADDR       TOTAL  NUM TEXT/RECORD  ATTRIBUTES  LINK ADDR/NUM STMT  RLD DISP  ENTRY ADDR  PROG SIZE  MRT LEVEL

 S  ORDHDR     221881/036289     17/0011      96/60        000000        170/00AA                                          3

0001 H                                                    05                  ORDHDR
0002 FSCREEN  CP  F    1000              WORKSTN
0003 FCMAST   IC  F 128 128R 6AI     4 DISK
0004 FSMAST   IC  F 128 128R 6AI     4 DISK
0005 FTRANS   UC  F 128 128R         DISK
0006 ICMAST   MA  11   1 CM   2 CA   3 C
0007 I                                          1   2 CRECCD
0008 I                                          3   3 CDELET
0009 I                                          4   90CUSNO
0010 I                                         10  34 CNAME
0011 I                                         35  59 CADDR
0012 I                                         60  81 CCITY
0013 I                                         82  83 CSTATE
0014 I                                         84  88OCZIPCD
0015 I                                         89  900CSLSNO
0016 ISMAST   SA  12   1 CS   2 CA   3 C
0017 I                                          1   2 SRECCD
0018 I                                          3   3 SDELET
0019 I                                          4   90SCUSNO
0020 I                                         10  34 SNAME
0021 I                                         35  59 SADDR
0022 I                                         60  81 SCITY
0023 I                                         82  83 SSTATE
0024 I                                         84  88OSZIPCD
0025 ITRANS   QU  01
0026 I                                          1   2 ORECCD
0027 I                                          3   50RR#1
0028 I                                          6   80RR#2
0029 I                                          9  110RR#3
0030 I        CU  15   1 CC   2 CU   3 C
0031 I                                          1   2 ORECCD
0032 I                                          3   3 ODELET
0033 I                                          4   90CUSNO
0034 I                                         10  150ORDNO
0035 I                                         16  40 CNAME
0036 I                                         41  65 CADDR
0037 I                                         66  87 CCITY
0038 I                                         88  89 CSTATE
0039 I                                         90  940CZIPCD
0040 I                                         95  960CSLSNO
0041 I                                         97 106 CPONO
0042 I        CS  16   1 CC   2 CS   3 C
0043 I                                          1   2 RCODE
0044 I                                          3   3 ODELET
0045 I                                          4   90CUSNO
0046 I                                         10  150ORDNO
0047 I                                         16  40 SNAME
0048 I                                         41  65 SADDR
0049 I                                         66  87 SCITY
0050 I                                         88  89 SSTATE
0051 I                                         90  940SZIPCD
0052 I        IT  17   1 CI   2 CT   3 C
0053 I                                          1   2 ORECCD
0054 I                                          3   3 ODELET
0055 I                                          4   90CUSNO
0056 I                                         10  150ORDNO
0057 I                                         16  170OLINE
0058 I                                         18  230ITMNO
0059 I                                         24  43 IDESCR
0060 I                                         44  490OQTY
0061 I                                         50  552OPRICE
0062 I                                         56  632OAMT
0063 I                                         64  68 IWHLOC
0064 I             18
0065 I* FORMAT- ORDSE1
0066 ISCREEN       01   1 CF   2 C1
0067 I                                   00010002 SCODE
0068 I                                   00030008 CUSNO
0069 I                                   00090014 ORDNO
0070 I* FORMAT- ORDSE2
0071 I             02   1 CE   2 C2
0072 I                                   00010002 SCODE
0073 I                                   00030008 CUSNO
0074 I                                   00090014 ORDNO
0075 I                                   00150039 CNAME
0076 I                                   00400064 SNAME
0077 I                                   00650089 CADDR
0078 I                                   00900114 SADDR
0079 I                                   01150136 CCITY
0080 I                                   01370138 CSTATE
0081 I                                   01390143 CZIPCD
0082 I                                   01440165 SCITY
0083 I                                   01660167 SSTATE
0084 I                                   01680172 SZIPCD
0085 I                                   01730182 CPONO
0086 I                                   01830184 CSLSNO
0087 I             10
0088 I* FORMAT- ORDSE3
0089 I                                   00010002 SCODE
0090 I                                   00030008 CUSNO
0091 I                                   00090014 ORDNO
0092 I                                   00150039 CNAME
0093 I                                   00400064 SNAME
0094 I                                   00650089 CADDR
0095 I                                   00900114 SADDR
0096 I                                   01150136 CCITY
0097 I                                   01370138 CSTATE
0098 I                                   01390143 CZIPCD
0099 I                                   01440165 SCITY
```

Figure 5-30 (Part 1 of 2). Partially Coded ORDHDR Program

```
0100 I                                          01660167 SSTATE
0101 I                                          01680172 SZIPCD
0102 I                                          01730182 CPONO
0103 I                                          01830184 CSLSNO
0104 I                                          01850186 ELINE
0105 I                                          01870192 ITEMNO
0106 I                                          01930198 QTY
0107 I                                          01990218 DESCR
0108 I                                          02190224 PRICE
0109 I                                          02250232 AMOUNT
0110 C* IF SCREEN E1, START, CHECK FOR CANCEL
0111 C   01                EXSR BEGIN
0112 C* IF SCRREN E2, WRITE RECORD AND SETON LR
0113 C   02                EXSR WRITE
0114 C*
0115 C           BEGIN     BEGSR
0116 C           CUSNO     CHAINCMAST                99
0117 C                     ENDSR
0118 C*
0119 C           WRITE     BEGSR
0120 C           CUSNO     CHAINSMAST                21
0121 C                     ENDSR
0122 O* PUT OUT SCREEN E1 IF SCREEN E1 WAS READ AND THERE IS AN ERROR, 99 ON
0123 OSCREEN  D         01 99
0124 O                                    K8 'ORDSE1  '
0125 O                             CUSNO  0006
0126 O                             ORDNO  0012
0127 O                         99  0060 '     INVALID          '
0128 O* PUT OUT SCREEN E2 IF SCREEN E1 WAS READ AND CK1 WAS PRESSED, KA ON
0129 O         D           01 KA
0130 O                                    K8 'ORDSE2  '
0131 O                             SCODE  0002
0132 O                             CUSNO  0008
0133 O                             ORDNO  0014
0134 O                             CNAME  0039
0135 O                             SNAME  0064
0136 O                             CADDR  0089
0137 O                             SADDR  0114
0138 O                             CCITY  0136
0139 O                             CSTATE 0138
0140 O                             CZIPCD 0143
0141 O                             SCITY  0165
0142 O                             SSTATE 0167
0143 O                             SZIPCD 0172
0144 O                             CPONO  0182
0145 O                             CSLSNO 0184
0146 O* PUT OUT SCREEN E3 WHEN ON SCREEN E1 CK1 WAS NOT PRESSED, OR FROM E2
0147 O         D           01NKA
0148 O        OR           02
0149 O                                    K8 'ORDSE3  '
0150 O                             CUSNO  0006
0151 O                             ORDNO  0012
0152 O                             CNAME  0037
0153 O                             SNAME  0062
0154 O                             CADDR  0087
0155 O                             SADDR  0112
0156 O                             CCITY  0134
0157 O                             CSTATE 0136
0158 O                             CZIPCD 0141
0159 O                             SCITY  0163
0160 O                             SSTATE 0165
0161 O                             SZIPCD 0170
0162 O                             CPONO  0180
0163 O                             CSLSNO 0182
0164 O                             ELINE  0184
0165 O                             ITEMNO 0190
0166 O                             QTY    0196
0167 O                             DESCR  0216
0168 O                             PRICE  0222
0169 O                             AMOUNT 0230
0170 O*                        99  ERRMSG 0278
```

Figure 5-30 (Part 2 of 2). Partially Coded ORDHDR Program

Coding with COBOL

You can also use COBOL (Common Business Oriented Language) to read data from and write data to a display station. You define the constants and fields that appear on the display screen with display screen format specifications. You can either enter the display screen format specifications explicitly or generate these specifications by using the Screen Design Aid. COBOL also allows your program to pass data to and read data from another application program through the use of the Interactive Communications Feature (SSP-ICF). Your program can communicate with a program running on the same System/34 or with a program running on another system. For more information on the SSP-ICF feature, refer to Chapter 2 in this manual or to the *Interactive Communications Feature Reference Manual*. For more information on using COBOL to read and write data from a display station, refer to *Transaction File Considerations* in the *COBOL Reference Manual*.

IBM System/34 COBOL-Supplied Procedures

IBM supplies several library procedures for use with System/34 COBOL. When the operator enters an appropriate command statement, the IBM-supplied library procedure is either executed or placed on the input job queue. For information on how to place a job on the input job queue, refer to the *Operator's Guide*.

Library procedures provide for COBOL compilation and link-editing, execution of COBOL programs, movement of a diagnosed source file to a library, and screen prompts, by using the following command statements:

- COBOL–compile a COBOL program.
- COBOLCG–compile and execute a COBOL program.
- COBOLG–execute a COBOL program with user-provided procedure for additional OCL statements.
- COBSYSIN–compile and link-edit a COBOL program entered from the current SYSIN device.
- COBMOVE–move a COBOL diagnosed source file to a library.
- COBOLP–provide screen prompts for entering, compiling, executing, and correcting COBOL programs.

TESTING THE PROGRAMS

To test the programs, procedures are built using SEU. The procedure name to execute a program is called by the same name as the program.

```
ORDHDR

// PROMPT MEMBER-ORDHDRFM,FORMAT-ORDSET,DATA-YES
// LOAD ORDHDR
// FILE NAME-CMAST,DISP-SHR
// FILE NAME-SMAST,DISP-SHR
// FILE NAME-TRANS,DISP-SHR
// RUN

ORDITM

// ATTR MRTMAX-3
// LOAD ORDITM
// FILE NAME-TRANS,DISP-SHR
// FILE NAME-TRANSLOG,DISP-SHR
// FILE NAME-CMAST,DISP-SHR
// RUN

ORDPRT

// LOAD ORDPRT
// FILE NAME-TRANSLOG,DISP-SHR
// RUN
```

The entire order entry procedure will be called by the procedure named ORDERS.

```
ORDERS

* BUILD TRANS FILE IF NOT PRESENT
// IFF DATAF1-TRANS BLDFILE TRANS,D,RECORDS,500,128
// IFF DATAF1-TRANSLOG BLDFILE TRANSLOG,D,RECORDS,2000,128
* EXECUTE ORDER HEADER PROGRAM, AN SRT
ORDHDR
* THEN THE LINE ITEM PROGRAM, AN MRT
ORDITM
* RELEASE THE TERMINAL AND CALL PRINT PROGRAM, AN MRT
// ATTR RELEASE-YES
ORDPRT
// IF SWITCH1-1 CANCEL
// RESET ORDERS
```

Before executing any procedure, the files used by each procedure must be on the system. DFU is used to build sample master files. Only a very small sample is entered.

Because the TRANS and TRANSLOG files are new files, they could be built using:

```
BLDFILE TRANS,D,RECORDS,500,128
BLDFILE TRANSLOG,D,RECORDS,2000,128
```

These statements are part of the final order entry procedure, ORDERS.

Considerations for Program Testing

Program testing is an important factor in the development of your application. Program testing is the process of running a program with the intent of finding errors in the program. By testing your program, you can verify if:

- Your program is correctly accepting input data.
- Your program is correctly processing the input data.
- Your program is generating output correctly, whether the output is a file, screen format, or a report.
- Your program can handle unexpected and erroneous data.

You should be aware of the following considerations when you begin to test your programs.

Adequate Time for Testing

Allow plenty of time for testing your programs. If you have a complex program, you may need more time than you did to write the program.

Test Schedule and Testing Logbook

Documenting the testing process is important. You should have a written record of the level of testing that your program was subject to. When you document your testing, be sure you have, for all cases, a description of the test data used as input to the program and a description of the correct output you expected for the test data. You should also assign some unique identification to each task in the testing process to aid you in the logging of the testing process.

Manageable Tasks

Divide the testing process into manageable tasks that proceed from the simple to the complex in terms of the functions performed in the program. Each task should build upon the functions of the preceding task. For example, if your program has to read three different record types, you should test to see if the program can read one, two, and then all three record types. Generally, the first task you should perform is an inspection of the code contained in the program for errors in logic or syntax.

Appropriate Test Data

Test your programs with data that checks whether the program is correctly performing its intended functions. In most cases, your program cannot be tested in a single test because certain functions must be tested individually. You must also check the way these functions interconnect in your program. Label your test data for each test, and retain this data for future use in the event that you have to modify your program due to errors found in it during testing.

Your test data should cover the following conditions:

- Normal and expected data
- Error and unexpected conditions

Normal and Expected Data

When you test normal and expected conditions, use data that is representative of the real data that you will be using in the program. For example, in testing an order entry application, data could be used to test the following conditions:

- Opening of a new account
- Updating an existing account
- Closing an existing account
- Updating multiple existing accounts
- Generating a picking slip report

Error and Unexpected Conditions

Use data in testing for error and unexpected conditions that is not representative of the data that you would use in running your program. Error data is important because you may discover that your program is generating errors when used in a new or unexpected way. By using data that is erroneous or unexpected, you will have a way of seeing whether your program performs predictably. Some examples of error condition testing are:

- Attempting transactions against nonexistent account numbers.
- Updating closed accounts.
- Using input data with invalid dates, incorrect totals, or invalid ranges of values in key fields.
- Using combinations of data that have multiple errors or data that has a combination of valid and invalid values.

Some examples of unexpected condition testing are:

- Using no data as input to a program
- Running two days' worth of data as one day's
- Running a program with the wrong inputs

It is important when you are testing error conditions to ensure that your program is issuing error messages that describe the errors encountered in testing.

Other Testing Considerations

Check the following when you are testing a program:

- Have you successfully tested the restart or recovery procedures?
- Are the personnel who have to use the program familiar with the procedures needed to use or run the program?
- Does the program pass the right data to other programs that have to use the program's output?
- Are the system operators familiar with the requirements for the program?
- Has the program been tested for all phases of processing? For example, if the program has to generate weekly and monthly reports, have you merely tested for weekly report generation? That is not a sufficient level of testing.

Program coding and testing should continue until all program functions are coded and tested. Using the DEBUG operation in complicated programs can shorten testing. If DEBUG is used, condition each DEBUG operation with an external indicator (for example, U8). Thus, the program can be tested with or without debugging without recompiling it.

As each program is being developed, changes may have to be made to the original specifications. For example, it may be easier to keep a field of information on the screen rather than in the control record. This would mean changing the screen and the logic of the program.

DOCUMENTING THE APPLICATION PROGRAM

This section provides some guidelines that you can use when documenting your application programs. Documentation of your programs can serve to provide the following functions:

- Definition of the purpose of the application program.
- Definition of the inputs to the program, such as screen formats, type of records, and organization of the inputs. (Are the inputs sequential on disk or diskette? Is the input passed from another application program?)
- Definition of the outputs from the program. (Is the output in the form of a screen format? Is the output a file on disk or a report and if so what is the format of the file or report?)
- Written documentation of the logic of the current version of the program.
- A history of the revisions that have been made to the program.
- Program accounting information.
- List of persons to contact for further information regarding the application program.

When documenting your program(s), you should establish a program packet that is readily accessible to persons who have to work with the application program. This program packet should be in a central location. Figure 5-31 shows items that you could place in your program packet.

- General program descriptions
  - Program name
  - Run frequency
  - General input descriptions
  - General output descriptions
  - Purpose of program
- Flow diagram of input and output to the program
- Detailed record layouts of input and output to the program[1]
- Printer layout of output report(s)
- Screen format descriptions of program input and output
- Narrative description of program logic
- Program source listing
- Sample of output
- OCL to run the program, or procedures necessary to run the program
- Operator instructions
- Program modification history log
- Message identification codes used by the program
  - Description of what these codes mean
  - Procedures to follow when these codes occur
- Restart/recovery information
- Instructions for the distribution of output reports

[1]You may wish to have all record descriptions in a central location to facilitate revising these record descriptions and to make sure that each program has an up-to-date record description.

Figure 5-31. Sample Program Packet List

The following eight pages make up a sample program packet for the ORDPRT program.

COMPUTER PROGRAM DESCRIPTION SHEET

Documentation Reference	Program Name	Library	Account Number	Version	Date
OR-03	ORDPRT	ORDLIB	11801	01	07/29/80

Language/Utility	Run Frequency	Input Format	Input From Program
RPG	DAILY	☒ Disk	ORDITM

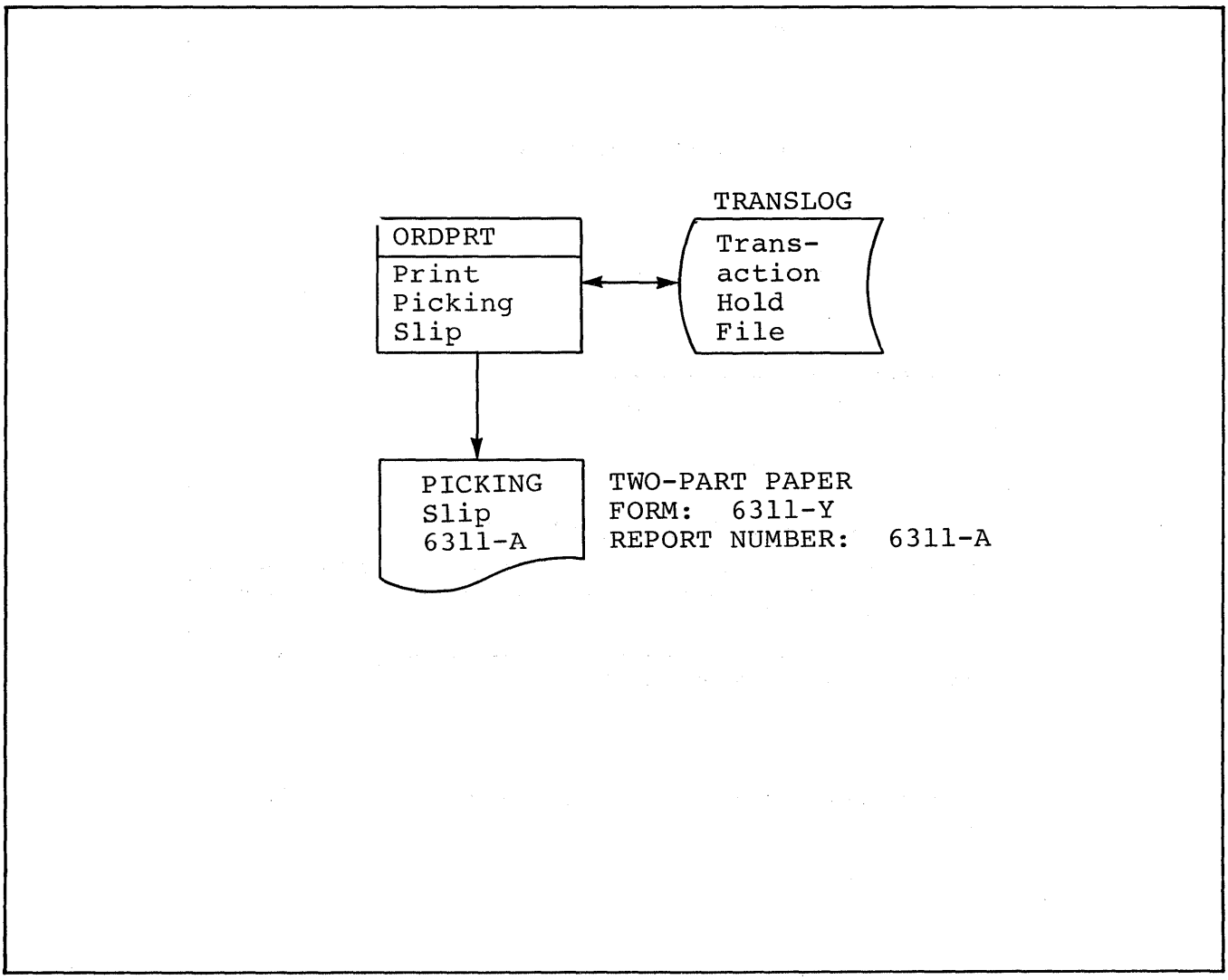
Input Screen Formats				

Output Report Number	Output Report Name	Output File Name	Output Used by Program Name	Output Used by Program Documentation Reference
6311-A	PICKING SLIP	TRANSLOG		

User Contact	Department
A G SMITH	INVENTORY

Program Used by	Responsible Programmer
ORDER ENTRY, WAREHOUSE DEPARTMENTS	AG SMITH

GENERAL FLOW DIAGRAM



PAGE 1 OF _____

COMPUTER PROGRAM DESCRIPTION SHEET

Documentation Reference	Program Name	Date
OR-03	Print Picking Slip (ORDPRT)	08/01/80

Program Segment Purpose

Narrative Description:

The purpose of the print picking slip program is to produce the picking slip by reading the TRANSLOG file and, for each order contained in the file, to produce a picking slip.

PAGE _____ OF _____

COMPUTER PROGRAM DESCRIPTION SHEET

Documentation Reference	Program Name	Date
OR-03	Print Picking Slip (ORDPRT)	08/01/80

Program Segment INPUT

NARRATIVE DESCRIPTION

1. TRANSLOG File (Transaction Hold File)

   File Organization: Direct

   Record Length: 128 characters

   Sequence: Record Code, Customer Number

PAGE _____ OF _____

INPUT/OUTPUT Record Description

File No. _____

Record Name TRANSACTION HOLD System SYSTEM/34 Page _____ of _____

File Name Translog ☒ Disk ☐ Diskette Date _____

File Organization Direct Sequence Record Code, Customer Number Prepared by A. Smith

Record Length 128 Key Customer Number Key Length 6

Created by ORDITM Used by ORDITM, ORDPRT Updated by ORDPRT

Values	Field Description	Field Name	Length	Decimal Pos.	Format	Location	
						From	To
	Relative Record Number		128		N	1	128
*Customer Record Information							
CV	Record Code	OCODE	2		A	1	2
D or blank	Delete Code	ODELET	1		A	3	3
	Customer Number	CUSNO	6		N	4	9
	Order Number	ORDNO	6		N	10	15
	Customer Name	CNAME	25		A	16	40
	Customer Address	CADDR	25		A	41	65
	City	CCITY	22		A	66	87
	State	CSTATE	2		A	88	89
	Zip Code	CZIPCD	5		N	90	94
	Salesman Number	CSLSNO	2		N	95	96
	Purchase Order Form	CPONO	10		A	97	106
*Ship to Record Information							
CS	Record Code	OCODE	2		A	1	2
	Delete Code	ODELET	1		A	3	3
	Customer Number	CUSNO	6		N	4	9
	Order Number	ORDNO	6		N	10	15
	Ship-To Name	SNAME	25		A	16	40
	Ship-To Address	SADDR	25		A	41	65
	City	SCITY	22		A	66	87
	State	SSTATE	2		A	88	89
	Zip Code	SZIPCD	5		N	90	94
*Line Item Record occurs 1 to 3 times for each Customer Record							
	Record Code		2		A	1	2
	Delete Code		1		A	3	3
	Customer Number		6		N	4	9
	Order Number		6		N	10	15
	Order Line Number		2		N	16	17
	Item Number		6		N	18	23
	Item Description		20		A	24	43
	Quantity Ordered		6		N	44	49
	Price		6		N	50	55
	Amount Extended		8		N	56	63
	Warehouse Location		5		A	64	68

COMPUTER PROGRAM DESCRIPTION SHEET

Documentation Reference	Program Name	Date
OR-03	PRINT Picking Slip (ORDPRT)	08/01/80

Program Segment Switch Settings and Local Data Areas

Narrative Description:

Switch Settings Used: None

Local Data Area Use: None

PAGE _____ OF _____

COMPUTER PROGRAM DESCRIPTION SHEET

Documentation Reference	Program Name	Date
OR-03	Print Picking Slip (ORDPRT)	08/01/80

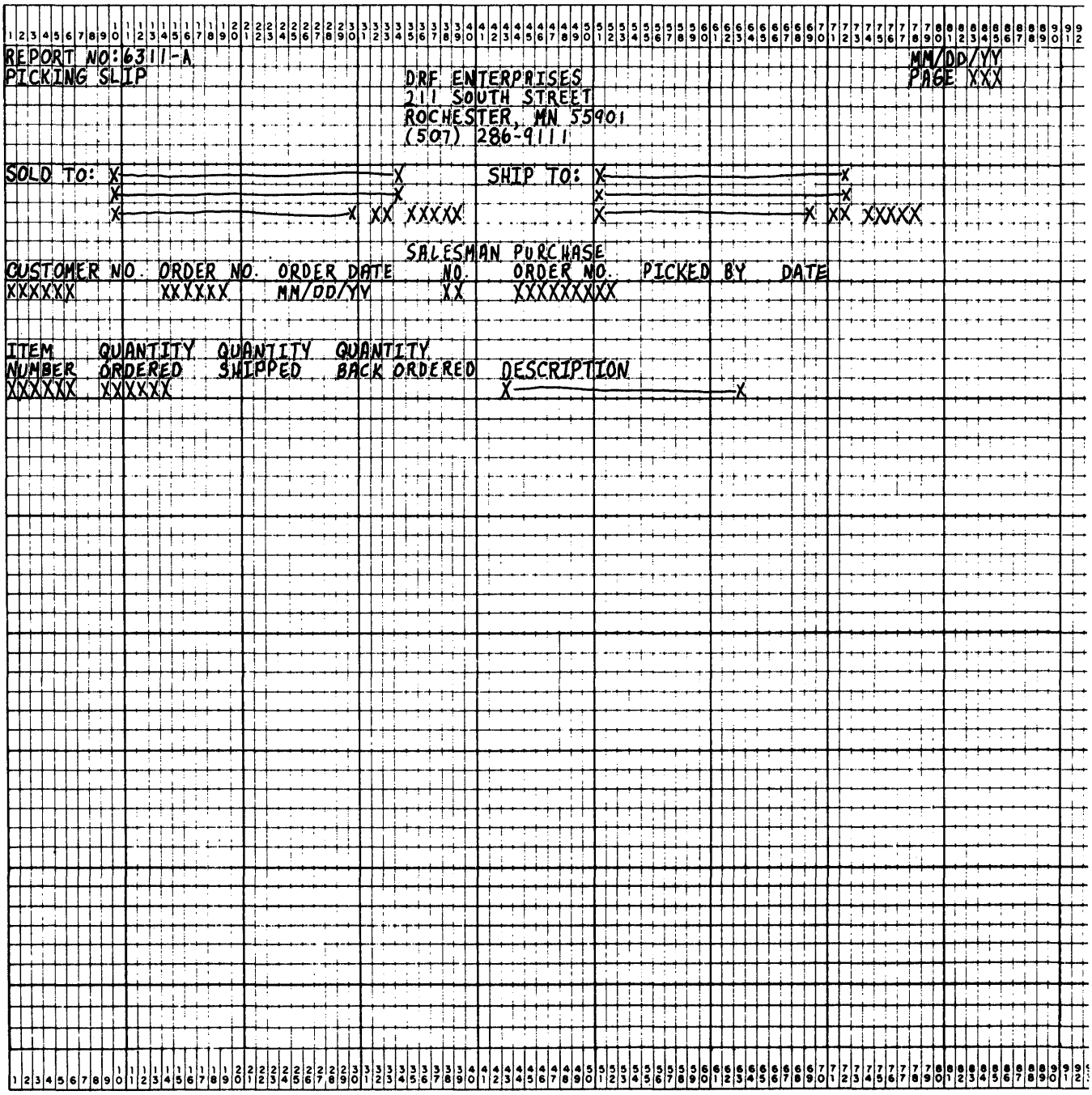
Program Segment OUTPUT

Narrative Description:

1. Picking Slip Report
   Form Number: 6311Y
   Report Number: 6311-A
   Two-Part Paper
   Frequency: Daily

2. TRANSLOG File
   File Organization: Direct
   Record Length: 128 characters
   Sequence: Record Code, Customer Number

PAGE _____ OF _____


REPORT NO: 6311-A
MM/DD/YY
PICKING SLIP
PAGE XXX
DRF ENTERPRISES
211 SOUTH STREET
ROCHESTER, MN 55901
(507) 286-9111
SOLD TO: X
X
X
XX XXXXX
SHIP TO: X
X
X
XX XXXXX
SALESMAN PURCHASE
CUSTOMER NO. ORDER NO. ORDER DATE NO. ORDER NO. PICKED BY DATE
XXXXX XXXXX MM/DD/YY XX XXXXXXXX
ITEM QUANTITY QUANTITY QUANTITY
NUMBER ORDERED SHIPPED BACK ORDERED DESCRIPTION
XXXXX XXXXX
X
X

COMPUTER PROGRAM DESCRIPTION SHEET

Documentation Reference	Program Name	Date
OR-03	Print Picking Slip (ORDPRT)	08/01/80

Program Segment Processing Logic

Narrative Description:

1. Open input file and outputs.
2. Read the control record of the TRANSLOG file to find the last record printed.
3. Read each record for each order in the file and, for each record, produce a line on the picking slip report (6311-A).
4. At the end of the order, update the control record with the relative record number of the last record printed.

PAGE _____ OF _____

System Test

When each application program has been tested and is working properly with minimum data, the application is ready for a system test.

The system test is the final phase in the development of an application. The system test makes sure that the application meets the objectives it was designed for and tests whether the system can be run as operational. During system testing you can test for the following items:

- Are the programs easy to use?
- Are the error messages provided by the programs meaningful?
- Is the final output correct?
- Does the system run as fast as it was intended to on the computer?
- Does the system meet its goals in terms of response time for display station operations?
- Do the recovery or restart functions work correctly?

By the end of the system test, you should have detailed instructions for the system operator and a set of instructions for the people who have to use the application.

User's Run Book

You should develop a user's run book after your application has been tested and implemented. A user's run book is written for people who have to use the application but were not involved in programming the application. The user's run book explains how the application works and gives detailed instructions for using the application.

Some of the items you should have in a user's run book are:

- A general description of the application and how the application works
- A description of the input to the application
- A description of the output of the application
- A detailed description of the procedures required to run the application
- Samples of all forms or documentation used by the application
- Error correction procedures

Operator Program Run Book

The operator program run book contains the procedures to be carried out when you run the application program on the computer. You should have procedures to run every program so that the system operator can obtain the proper inputs to the program, set up the inputs correctly, and make sure the job executes correctly. Generally, entries in an operator program run book can be contained on one 8 1/2 by 11 inch sheet of paper and should contain information necessary for proper execution and setup of the program.

Sample items that can be included in an operator program run book are:

- A system flowchart of the application
- A list of the inputs used by the program
- Procedures for the setup of the job
- Procedures for error handling
- Procedures describing output distribution that can be used by either the system or subconsole operator
- Procedures for restarting or rerunning the job

Figure 5-32 is a sample of an operator program run book entry.

OPERATOR INSTRUCTIONS

PROGRAM NAME: PR85			LIBRARY USED: PAYROLL		
PROGRAM PURPOSE: This program is responsible for writing a Deduction Register					
	FILE NAME	FILE MEDIUM	DISK/DISKETTE/ LABEL	DRIVE	DISPOSITION AFTER USE
FILES USED	DEDUCT	DISK	N/A	N/A	REMAINS ON DISK

PRINTER USED	FORMS USED	DISTRIBUTION
SYSTEM	3-PART STOCK	Original and one copy to payroll; one copy remains in the file.

RUN FREQUENCY	PREREQUISITE PROGRAM	NEXT PROGRAM
BIWEEKLY as a part of each payroll cycle.	NONE	PR87

INSTRUCTIONS REQUIRED TO RUN THE PROGRAM

```
// LOAD PR85
// FILE NAME-DEDUCT          OR
// RUN
```

```
// MENU DEDUCT, PAYROLL
USE OPTION 6
```

EXPECTED ERRORS/RECOVERY

UNIDENTIFIED RECORD IN THE FILE—take recovery option 1 and continue

UNEXPECTED ERRORS/RECOVERY

WRITE DOWN ERROR MESSAGE CODE, THE MESSAGE TEXT, AND TAKE RECOVERY OPTION 0 OR 1. IF 0 OR 1 ARE NOT VALID OPTIONS, CONTACT THE PROGRAMMER OR MANAGER.

OPERATOR NOTES:

COMPARE CONTROL TOTALS TO THOSE IN THE CONTROL BOOK... ALERT PAYROLL IF THEY DON'T BALANCE & DON'T RUN PR87

```
COMMAND
                                        MENU: DEDUCT
 1. JOURNAL ENTRIES          (PR80)      13.
 2. INQURIY-JOURNAL FILE     (PR81)      14.
 3. MASTER FILE LISTING      (PR82)      15.
 4. EDIT TIME CARDS          (PR83)      16.
 5. ALLOWABLE VARIANCE       (PR84)      17.
 6. DEDUCTION REGISTER       (PR85)      18.
 7. PAYROLL CHECKS           (PR86)      19.
 8. PAYROLL REGISTER         (PR87)      20.
 9. COPY PAYROLL FILE        (PR88)      21.
10.                                      22.
11.                                      23.
12.                                      24.

ENTER NUMBER, COMMAND, OR OCL.
 6
```

Figure 5-32. Operator Program Run Book Entry

Appendix A. Display Station Operations Requested by Basic Assembler Programs

This appendix describes the basic operations that can be requested by a Basic Assembler program when it calls work station data management. When the program calls work station data management, two bytes in a data area called the DTF (define the file) determine the requested operation. These two bytes can request a combination of the basic operations. An example is the *put-no-wait with invite* operation. This operation is a combination of the put-no-wait and invite input operations.

The following information lists and describes the operations that can be requested by a Basic Assembler program when it calls work station data management. The macro instruction, operation code, or operation modifier that is used to perform an operation is listed in parenthesis beneath the operation. Refer to the *Basic Assembler and Macro Processor Reference Manual* for further information.

Operation	Description
Open ($OPEN)	Prepares for the processing of all display station operations requested by the user program. During the open operation, work station data management builds an index that points to all display screen formats in the display screen format load member used by the program. If specified in the DTF, work station data management builds a table containing information supplied on the WORKSTN OCL statements. In Basic Assembler programs, you must request an open operation before you request any other operations for the display station file.
Put (PUT)	Sends data to a specified display station. Work station data management returns control to the user program when the data transfer is complete.
Put-no-wait (PNW)	Sends data to a specified display station. Control returns to the user program when the operation is scheduled. The program can then change its record area, indicators, or DTF without affecting the scheduled operation. If the program attempts a second put-no-wait operation to the same display station, work station data management waits for the first operation to be completed before it schedules the second operation.

Operation	Description
Put for read-under-format (PRUF)	Sends data to a display station and allows the next job step to read the data back in. Two types of PRUF operations can be requested: • Auto PRUF, which can only be requested from a Basic Assembler program. An auto PRUF operation uses the display screen to temporarily store data between job steps. The keyboard is locked when the format is displayed. The format is returned to the next job step in response to the first accept input operation requested from the step. To perform an auto PRUF operation, the Basic Assembler program must request an auto-PRUF-put-no-wait-invite operation. • Interactive PRUF. An interactive PRUF operation occurs if the last display station operation before termination of a job step is an invite type operation. The keyboard is unlocked for an interactive PRUF operation, and the operator can key information while the next step is loading. After the operator enters the display, work station data management returns the format to the next job step in response to the first accept input operation requested from the job step. *Note:* Either PRUF operation can occur when going from (1) an MRT program to another MRT program, (2) an MRT program to an SRT program, (3) an SRT program to an MRT program, or (4) an SRT program to another SRT program. After requesting a PRUF operation, an MRT program should release the display station. After requesting a PRUF operation, an SRT program should end.

Operation	Description
Put override (OVR)	Sends data to a specified work station but transmits only indicator-controlled output fields and attributes. A program uses a put override operation to override (change) information previously displayed without retransmitting the entire format. During a put override operation, the following events occur: • If a field had an indicator specified in the output data entry (columns 23 and 24 of the D specification) and that indicator is off, the data in the field is unchanged. If data was entered in the field, that data is unchanged. Any field that had Y, N, or blank specified in columns 23 and 24 is also unchanged. • If a field had an indicator specified in the output data entry (columns 23 and 24 of the D specification) and that indicator is on, the field is retransmitted and contains data from the program. Any data that was entered in the field by the operator is lost. Output information is displayed from the *same location in the output record area* as for a normal put operation. • For all fields, the use of indicator-controlled attributes, such as highlight or reverse image, is determined by the state of that indicator. All field attributes that are not controlled by indicators are unchanged. *Note:* If an indicator is specified in the protect field entry (columns 37 and 38 of the D specification), that indicator is ignored during the override operation.
Unformatted request (UNF)	Modifies any of the put operations and informs work station data management that the information in the program's record area is in a format suitable for transmission to the display station. (The information already contains all the required control information and attribute characters.) Work station data management transmits the information to the display station without formatting the information.

Operation	Description
Invite input (INV)	Signals the display station that the program is ready to receive data. The invite input operation is in effect until (1) the program issues a get input operation for the display station, (2) an accept input operation receives data from the display station, (3) the program requests a stop invite operation for the display station and the stop invite is successful, or (4) the program requests an output operation to the display station. *Note:* A display station operator can interrupt an MRT program (by pressing the Attn key) only while an invite input operation is in effect for the display station.
Get (GET)	Receives data from a specified display station. After placing the data in the program's record area, work station data management returns control to the user program.
Accept (ACI)	Receives data from any display station that responded to a previous invite input operation. For example, if a program issues three invite input operations for three display stations (W1, W2, and W3) and then issues an accept input operation, the data the program receives can be from any one of those display stations. Work station data management places the ID of the display station that returned the data in the program's DTF.
Stop invite input (STI)	Cancels a previous invite input operation to a specified display station. If the display station operator enters the display before the stop invite input operation is performed, work station data management informs the user program via a return code in the DTF. The program can then request a get or accept operation to read the information entered by the operator, or the program can request an output operation to the display station. If the program requests an output operation, the information the operator entered is lost. *Note:* The user program does not have to issue a stop invite operation to override an invite input operation. For example, a program can issue two consecutive put with invite operations. However, any information the operator enters in response to the first invite operation is lost.
Acquire (ACQ)	Allocates the specified display station to the program that requests the operation. If the acquire operation fails, a return code is passed back to the user program.

Operation	Description
Release (REL)	Releases the specified display station from the program that requests the operation. A release operation is valid only from an MRT program.
Write error (ERROR)	Writes up to 78 high-intensity characters of data on the bottom line of the display screen. The keyboard locks, and the operator must press the Error Reset key to restore the previous contents of the bottom line.
Get attributes (GTA)	Places an 8-byte string of data in the user program's record area. The data string is a series of bytes that describes the attributes of the specified work station.
Extended get attribute (EGTA)	Places a 16-byte string of data in the user program record area. The data string is a series of bytes that describes the attributes of the specified ideographic-capable display station. *Note*: The get attributes (GTA) operation can also be used for an ideographic-capable display station, but some of the attributes for the specified display station will not be returned.
Print (PRINT)	Prints, on a specified printer, the contents of a specified display screen.
Roll (ROLL)	Rolls a specified group of lines up or down on the display screen. The user program's DTF specifies the starting and ending line numbers of the area to be rolled, the number of lines to roll, the direction of the roll, and whether or not vacated lines should be blanked out.
Erase (ERS)	Erases (blanks out) the contents of the input fields on the display. An erase operation instead of a put operation is performed when you specify erase input fields (columns 31 and 32 of the S specification). If an invite or get operation is not requested along with the erase operation, the keyboard is locked following an erase operation. Otherwise, the keyboard is reset.
Clear (CLR)	Erases (blanks out) the entire display screen, including attribute bytes. You can request the clear operation only from a Basic Assembler program.

Operation	Description
Status inquiry (SIQ)	Causes work station data management to set a DTF return code, which indicates: • Whether any invite input operations have completed. • Whether the system operator has used the STOP SYSTEM command to cause shutdown of system operations. *Note:* Status inquiry can also be requested by passing to work station data management a value of hexadecimal 'FFFF' in index register 2. Work station data management returns the status information in index register 2.

Glossary

abnormal termination: A system failure that does not allow an operator to sign off successfully. For example, an abnormal termination occurs after a message that has only a 3 option or after a failure that stops the system but does not cause a message to be displayed.

active program: A program that is in main storage or in the swap area on disk.

active program list: A system-maintained list of all active programs.

active user library: The library used by most functions performed during execution of a customer-written program. If a required member is not in the active user library, the system library (#LIBRARY) is searched for the member.

application: A particular data processing task such as inventory control or payroll.

assign/free area: The available space in the supervisor for control areas.

audit trail: A general recording of who did what and in what sequence.

autowriter: A function that causes the spool writer program to be loaded without operator intervention whenever output exists in the spool file.

BASIC (Beginners All-Purpose Symbolic Instruction Code): An interactive programming language designed for ease of use.

basic display station configuration record: A disk record that contains the basic display station configuration information on the IBM-supplied PID diskettes.

batch classification: A classification assigned to programs by the System/34 swapping function. A program is assigned the batch classification when it executes for longer than a system-determined time limit without accepting input from a display station. See also *interactive classification.*

batch processing: A method of running jobs that does not require continuous operator attention; that is, processing that is not interactive. Contrast with *interactive processing.*

block: A 10-sector unit of disk storage that contains 2560 bytes. A block is also a group of records, treated as a logical unit, that is read or written by the computer.

COBOL (Common Business Oriented Language): A standardized business language for programming a computer.

command display station: A display station that can request and initiate jobs.

command mode: A mode that a display station can be placed in. In command mode, a display station is capable of requesting jobs or initiating jobs.

command processor: The SSP function that initially processes information entered by the operator.

data communications configuration record: A disk record that contains information about the data communications configuration. Each command display station has an associated data communications configuration record.

DFU (Data File Utility): Part of the Utilities Program Product used to create, maintain, and display or print data files.

direct file: A disk file in which records are assigned specific record positions. Regardless of the order in which records are put in a direct file, they always occupy the assigned position in the file.

directory: See *library directory.*

disk data management: The SSP function that controls the flow of data to and from disk files.

dispatching function: The System/34 function that allows multiple programs in main storage to share processing time.

display screen format: A table that defines a display presented by work station data management.

display station local data area: A 256-byte area on disk that can be used to pass information between jobs and job steps during a session. A separate local data area exists for each command display station.

erase input operation: A work station data management operation that erases (blanks out) the contents of the input and output/input fields on the display screen.

external indicators: Eight indicators (U1-U8) that are normally set by the SWITCH OCL statement before job execution. These indicators can be tested and changed during execution.

fixed-format menu: A type of menu generated by the $BMENU utility program. A fixed-format menu contains two columns of menu items, with 12 items in each column.

FORTRAN (formula translation): A programming language primarily used to express arithmetic formulas via computer programs.

free-format menu: A type of menu generated by the $BMENU utility program. For a free-format menu, the programmer defines the contents of lines 3 through 20.

history file: An area on disk in which a log of specified types of system actions and operator responses is recorded.

IFILE: An attribute of an indexed file that allows sequential-by-key access to added records without sorting the keys.

initiator: The SSP function that reads and processes OCL statements from the system input device.

inquiry: 1. A request (entered from a display station) for information in storage. 2. A request for information that puts the system in inquiry mode. (The operator initiates an inquiry by pressing the Attn key.)

inquiry latch: An indicator that informs an interrupted program of an inquiry request. The operator causes the inquiry latch to be set on by selecting option 4 on the Inquiry display.

interactive classification: A classification assigned to programs by the System/34 swapping function. If a program executes for longer than a system-determined time limit without accepting input from a display station, the program loses its interactive priority. See *batch classification.*

interactive processing: A method of processing in which each operator action causes a response from the system or a program, as in an inquiry system or an airline reservation system. See *batch processing.*

IPL (initial program load): A sequence of events that loads the system programs and prepares the system for execution of jobs.

job: One or more related procedures or programs grouped into a first-level procedure.

job file: A disk file created with a retain parameter of J. A job file can be used by all the job steps in a job. The job file is defined only within the job and does not exist after the job ends.

job region: The amount of main storage ensured by the SSP for use by a job. The job region is specified by the REGION OCL statement, the SET procedure, or the $SETCF utility program.

job step: A unit of work represented by a single program. The LOAD OCL statement, RUN OCL statement, and other OCL and utility control statements define the job step within a procedure.

library: An area on disk that can contain load members, procedure members, source members, and subroutine members.

library control sector: The first sector in a library directory. The library control sector contains a record of the used and available space in the library.

library directory: A variable-sized area that contains information, such as name and location for each member in the library.

library member: A named collection of records or statements in a library.

line printer (5211 or 3262 printer): A device that prints all characters of a line in a single operation. Contrast with *serial printer, matrix line printer.*

linkage editor: A program that prepares subroutines or the output of language translators for execution. The linkage editor resolves symbolic cross-references, generates overlay structures on request, and produces executable code (a load member) that is ready to be loaded into main storage.

load member: A collection of instructions that the system can execute to perform a particular function, regardless of whether the function is requested by the operator or specified in an OCL statement. Load members can also contain screen formats and message members. Load members are stored in a library.

local data area: See *display station local data area.*

LR (last record) indicator: An RPG II indicator that signifies when the last data record is processed; the LR indicator is used to condition all operations that are to be done at the end of the program.

master file: A collection of permanent information, such as a customer address file, that is often processed along with a transaction file.

matrix line printer (5225): A device that prints all the characters of a line in a single operation, and each character is formed by a matrix of wires.

menu: A displayed list of items (usually jobs) from which the operator makes a selection.

memo updating: An interactive file updating technique.

MRT (multiple requestor terminal) procedure: A procedure that calls an MRT program.

MRT (multiple requestor terminal) program: A program that can process requests from more than one requesting display station concurrently.

multibatch processing: The processing of two or more batch programs concurrently.

NEP (never-ending program): A program that uses system resources for a long period of time and was defined as a never-ending program (NEP-YES) on the COMPILE OCL statement.

nonswappable storage: The amount of user storage used by nonswappable programs.

NRT (non-requestor terminal) program: A program that is evoked by another procedure or program or has no requesting display stations allocated to it. Contrast with *SRT* and *MRT.*

OCL (operation control language): A progamming language used to identify a job and its processing requirements to the SSP.

operator control command: A command statement used by an operator to control system or display station operation. A control command does not run a procedure and cannot be used in a procedure.

override fields operation: A work station data management operation that allows a program to override (modify) fields on a display without retransmitting the entire display.

password security: An optional SSP function that helps prevent the unauthorized use of a display station.

permanent file: A file that remains in existence until deleted by using the $DELET utility. A permanent file is created with a retain parameter of P for disk or 999 for diskette.

PID: Program information department. An IBM group responsible for distributing programs and publications.

print intercept routine: The routine that causes printer output to be placed in a spool file in disk storage rather than going to the printer.

print spooling: A part of the SSP that provides temporary storage of print data on disk.

printer data management: The SSP function that controls the flow of information to the printer.

priority processing: A method used in a multiprogramming environment that determines the sequence in which programs are processed by the system.

priority program: A program to which the priority attribute was assigned by the operator or programmer.

procedure: A set of related OCL statements and, possibly, utility control statements, that cause a specific function or set of functions to be performed. A procedure in a library is called a procedure member.

procedure command: A command statement that runs a procedure. A procedure command is a special form of the INCLUDE OCL statement.

procedure member: A procedure that is stored in a library.

production run: The normal operational running of an application system.

program ready list: A system-maintained list of programs that are in main storage and ready to execute.

put override: See *override fields operation.*

record mode: A method of operation in which data is transferred by the SSP one record at a time.

resource security file: A file that contains information about each protected file or library.

RPG II: A commercially oriented programming language specifically designed for writing application programs that meet common business data processing requirements.

scratch file: A file that can be used only by the job step creating it. A scratch file does not exist after the job step ends if the file is created with a retain parameter of S.

SDA (screen design aid): A part of the Utilities Program Product used to create, change, and delete display screen formats and menus. SDA can also be used to build RPG II programs and WSU programs.

sector: 1. A 256-byte area on disk reserved to record data. 2. The smallest amount of data that can be written to or read from a disk or diskette during a single read or write operation.

sector mode: A method of operation in which data is transferred by the SSP one sector at a time.

serial printer (5256 Printer): A printer that prints characters one at a time. Contrast with *line printer, matrix line printer.*

session: 1. The time during which programs or devices can communicate with each other. 2. The elapsed time that starts when an operator signs on the system and ends when the operator signs off the system.

SEU (source entry utility): A part of the Utilities Program Product used by the operator to enter, update, and print procedures and source programs in a library.

single program mode: A method of operation during which one job (either batch or interactive) is completely processed before another job is begun.

source member: A collection of records (such as RPG II specifications or sort sequence specifications) that is used as input for a program. Source members are stored in a library.

SRT (single requestor terminal) program: A program that can have only one requesting display station at a time.

SSP: System support program product.

standby mode: A method of operation in which a display station is waiting to be acquired and used by a program running on the system.

subroutine member: A subroutine that needs to be link-edited before being loaded for execution. Subroutine members are stored in a library.

SUBR22: An IBM-written subroutine that allows an RPG II program to read records from a transaction file created by a WSU program.

SUBR95: An IBM-written subroutine that allows an RPG II program to perform an inquiry function.

supervisor: A program that manages system resources such as the printer(s), display station(s), main storage, input job queue, and spooling.

suppress input operation: A work station data management operation that does not invite input from a display station.

swapping function: The System/34 function of placing programs or segments of programs temporarily on disk; swapping allows the total amount of user storage required by concurrently executing programs to exceed the amount of main storage normally available for user programs.

system library: The library containing members that are part of the SSP in addition to non-SSP members. The system library is labeled #LIBRARY and cannot be deleted from disk.

system list function: An SSP function that prints output for some SSP utility programs and service aids.

system printer: The printer, named at system configuration time, that is used for system and display station printed output unless the output is specifically directed to another printer.

temporary file: A file that cannot be automatically deleted until after its expiration date. A temporary file is created with a retain parameter of T for disk or 001 through 998 for diskette.

terminator: The System/34 function that performs the system action necessary to end a job or job step.

transaction file: A file containing relatively transient data that, for a given application, is processed together with the appropriate master file.

user storage: The area of main storage that is not used by the SSP.

work station data management: The SSP function that enables a program to present data on a display screen by providing only a string of data fields and a format name.

WSU (work station utility): A part of the Utilities Program Product that performs an interactive data entry and edit function.

Index

IBM System/34
Concepts and Design Guide

READER'S COMMENT FORM

SC21-7742-3

Please use this form only to identify publication errors or to request changes in publications. Direct any requests for additional publications, technical questions about IBM systems, changes in IBM programming support, and so on, to your IBM representative or to your nearest IBM branch office.

☐ If your comment does not need a reply (for example, pointing out a typing error) check this box and do not include your name and address below. If your comment is applicable, we will include it in the next revision of the manual.

☐ If you would like a reply, check this box. Be sure to print your name and address below.

Please contact your nearest IBM branch office to request additional publications.

Name _____

Company or Organization _____

Address _____

_____

City State Zip Code

IBM may use and distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

No postage necessary if mailed in the U.S.A.

Fold and tape Please do not staple Fold and tape

NO POSTAGE
NECESSARY IF
MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N. Y.

POSTAGE WILL BE PAID BY . . .

IBM CORPORATION
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold and tape Please do not staple Fold and tape

IBM System/34 Concepts and Design Guide (File No. S34-34) Printed in U.S.A. SC21-7742-3

SC21-7742-3