60494800

CONTROL DATA CORPORATION

SCED USER GUIDE FOR INTERCOM 4 MULTI-USER JOB CAPABILITY

CONTROL DATA®
CYBER 170 SERIES
CYBER 70 MODELS 72, 73, 74
6000 SERIES COMPUTER SYSTEMS

REVISION RECORD	
REVISION	**DESCRIPTION**
A	Original printing.
(11-01-75)	
Publication No. 60494800	

REVISION LETTERS I, O, Q AND X ARE NOT USED

Address comments concerning this manual to:

CONTROL DATA CORPORATION
Publications and Graphics Division
215 MOFFETT PARK DRIVE
SUNNYVALE, CALIFORNIA 94086

or use Comment Sheet in the back of this manual

LIST OF EFFECTIVE PAGES

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

Page	Revision	Software Feature Change
Cover	–	
Title Page	–	
ii thru viii	A	
1-1, 1-2	A	
2-1 thru 2-7	A	
3-1 thru 3-21	A	
4-1 thru 4-4	A	
A-1, A-2	A	
B-1, B-2	A	
Cmt Sheet	A	
Return Env	–	
Back Cover	–	

Page	Revision	Software Feature Change

PREFACE

his manual describes the SCED interface routines vailable for programming COBOL 4 multi-user jobs iat operate under INTERCOM 4.5 and NOS/BE 1 n CONTROL DATA® CYBER 170, CYBER 70 lodels 72, 73, and 74, and 6000 Series computers.

This manual is intended for use by an experienced COBOL application programmer who is not experienced in programming multi-user jobs. Other manuals that might be of interest to the user are listed below:

Publication	Publication Number
INTERCOM 4 Reference Manual	60494600
COBOL 4 Reference Manual	60496800
INTERCOM 4 Multi-User Job Capability Reference Manual	60494700
NOS/BE 1 Operating System Reference Manual	60493800

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

INTRODUCTION 1

CONCEPTS OF A MULTI-USER JOB

A multi-user job (muj) is a program that can be shared by a number of terminal users. The muj program is loaded into central memory and activated when a user types the program name as an INTERCOM command. Once a copy of the muj program is loaded into central memory, any other users who enter the name of the program are linked to the existing program.

One advantage of muj program processing is that the use of computer resources, such as central memory and disk storage, might be reduced because only one copy of a muj program is in execution at one time. The muj program differs from current INTERCOM programs, such as TEACH, which have a separate program in execution for each user requesting the program. If five users were executing TEACH at the same time, five copies of TEACH would be loaded into central memory; if five users were executing a muj program at the same time, only one copy of the muj program would be loaded into central memory.

Another advantage of muj program processing is that users can expect faster response time. A muj program is loaded into central memory when the first user enters the program name as an INTERCOM command; it is terminated when the last user executing in the program is disconnected. During the time a muj program is executing, it may be either swapped in or swapped out of central memory. A muj program that is widely used by a large number of terminals can stay swapped in for large periods of time. If the muj program is already swapped in at the time a user requests the program, the response time is greatly reduced.

SYSTEM/MULTI-USER JOB INTERFACE

Multi-user job subroutines are a set of INTERCOM library subroutines that provide the standard interface between INTERCOM and a muj program. Muj subroutines perform such tasks as scheduling, issuing system requests for input/output, and monitoring waiting users; muj subroutines can be called from COBOL, COMPASS, and FORTRAN Extended programs. A detailed discussion of these subroutines is contained in the INTERCOM 4 Multi-User Job Capability reference manual.

SCED is a set of interface routines that provide a higher-level interface between COBOL programs and muj subroutines. SCED contains a set of entry points that are called by a COBOL muj program. The COBOL muj program is written as though it were to service only one terminal; therefore, the author of the program does not have to be aware of, or program for, the individual identities of users of the program. SCED provides the mechanism for sharing the program by performing the appropriate functions when the COBOL muj program issues SCED calls. The SCED functions are:

- Scheduling users for processing and returning each user to the program at the point where the user last stopped executing.
- Maintaining and allocating buffers for terminal input/output.
- Communicating status information (and terminal input/output requests) to and from the muj subroutines through subroutine calls.
- Scheduling the execution of initialization and termination code in the muj program.

Figure 1-1 shows the organization of muj programs and the interface of the programs with the system. A COBOL muj program can communicate with SCED; SCED communicates with the muj program and muj subroutines; muj subroutines communicate with SCED and INTERCOM; and INTERCOM communicates with muj subroutines and the terminal users. The other muj programs shown communicate directly with muj subroutines.

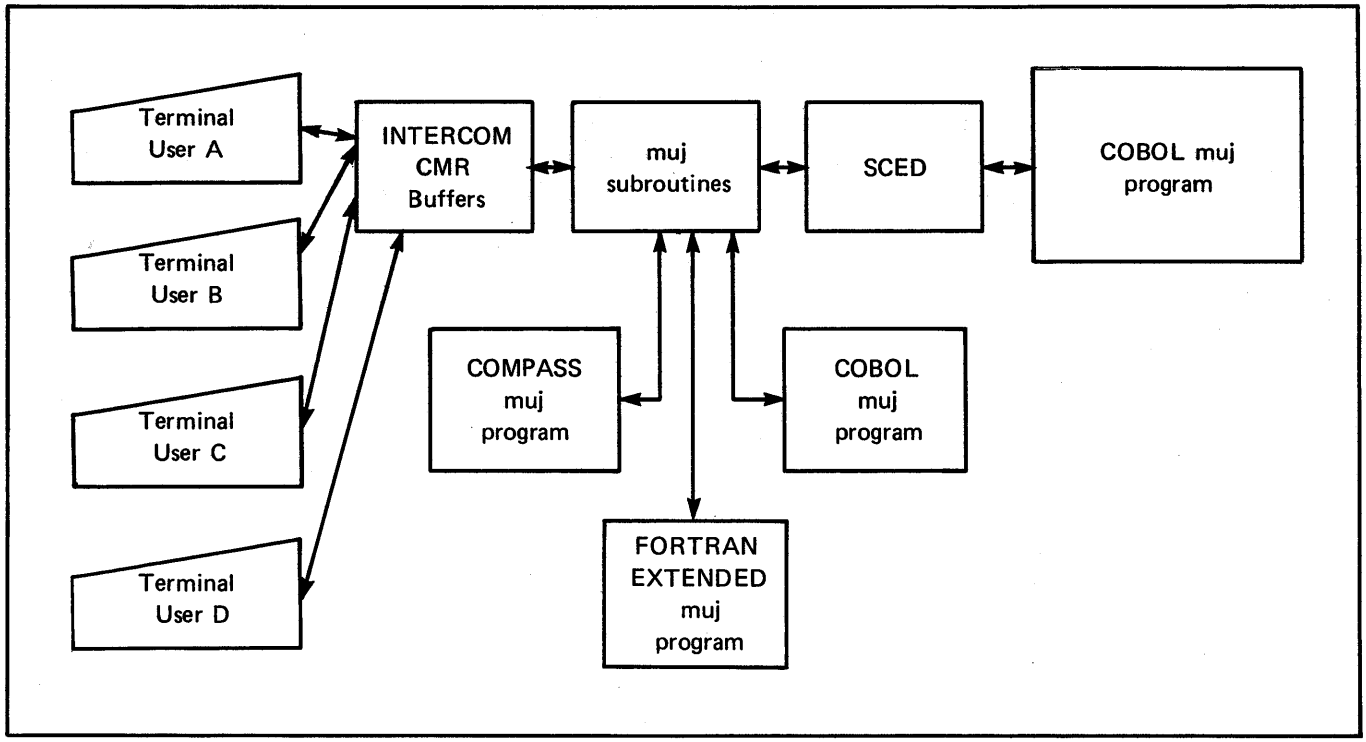

Figure 1-1. Muj Program Interface

PRINCIPLES OF WRITING A COBOL MUJ PROGRAM USING SCED

2

SCED/COBOL MUJ PROGRAM INTERFACE

SCED is a set of interface routines that provide the mechanism for sharing a COBOL muj program among terminal users. The COBOL muj program calls SCED to perform nine functions. The place in the COBOL muj program where a SCED function is called is known as a SCED point. The SCED calls are listed below.

SCED Call	Function
INIT	Initializes the muj program for execution.
CONNECT	Connects a user to the muj program.
TERMIN	Performs a terminal input operation.
TERMOUT	Performs a terminal output operation.
GETINT	Reserves an interlock.
RETINT	Releases an interlock.
IOWAIT	Relinquishes control of the muj program for a user when an input/output operation is initiated with the SEEK verb.
DISCON	Disconnects a user from the muj program.
EXITMUJ	Terminates the muj program.

SCED is designed to allow other users to execute in the muj program while some of the SCED functions are being performed for a given user. Program sharing is especially significant for SCED functions that are entered for terminal and disk input/output operations, which slow execution of a program considerably.

The first user to request execution of a COBOL muj program causes the program to be loaded into central memory and begin executing. As soon as the first user arrives at a SCED point that can relinquish control of the program, SCED stops processing for that user and starts processing for any ready users. Each ready user is connected in turn and advanced to a SCED point where program control is relinquished.

SCED maintains a table of users connected to the program and knows the stage of processing of each user at all times. When more than one user is connected to the program and the user executing reaches one of the SCED points where program control can be relinquished, SCED advances all ready users to their next SCED points before control returns to the original user. SCED schedules each ready user to continue processing in the program from the point at which the user last stopped processing, even though any number of users might have executed in the same or different areas of the program in the meantime. Thus, a general pattern of sharing emerges.

Most of the time each user is connected to the program, SCED is waiting for an input/output activity to be completed for the user. Input/output operations, such as terminal input, which must wait for the terminal user to enter data, can take an indeterminate amount of time for completion. Any number of ready users can advance to their next respective SCED points during the time it takes a terminal input operation to be completed, since processing time required to proceed along a path from SCED point to SCED point is typically no more than a few milliseconds.

SCED overlaps the functions of terminal input/output for one user with processing for other users by maintaining buffers for terminal input/output operations. Since the actual receipt or transmittal of terminal input/output takes place in the SCED buffers, the program can continue in execution for other users during the time it takes the terminal input/output operation to be completed.

SCED is designed primarily for applications that require on-line program access to a data base. All accesses to a data base in a muj environment are essentially random, even if one user requires sequential access to part of a file. A call to the SCED function IOWAIT can be made immediately after random file access is initiated with the SEEK verb. This SCED call allows other users to execute in the muj program while the SCED function is being performed for a user.

The SCED interface does not provide a facility for creating or attaching files. All files accessed through a COBOL muj program by terminal users must be attached and returned by calls to subroutines that are written in languages that can reference RA+1. The files must reside on nonallocatable devices.

ORGANIZING A COBOL MUJ PROGRAM

The Procedure Division of a COBOL muj program can be divided into three operational parts for discussing the program's general sharing techniques: initialization, terminal session, and wrap-up. The initialization code initiates processing of a COBOL muj program; the terminal session code contains all the instructions that are to be shared by users of a muj program; and the wrap-up code terminates processing of a muj program.

INITIALIZATION

The initialization section is the first section of a COBOL muj program. The code in this section is executed when the first user calls the program through an INTERCOM command. Users connecting to a running muj program do not execute the initialization section. Housekeeping functions that are executed only once for each activation of the program are performed in this section. The housekeeping functions should include attaching and opening files, initializing counters, and any other functions that are necessary to ready the program for processing.

TERMINAL SESSION PROCESSING

The terminal session section, the second section of a COBOL muj program, is the only section that is shared. Any user connecting to a program already in execution begins processing at the beginning of this section. The section must therefore include all elements normally contained in the Procedure Division of a single user job except those operations, such as opening and closing shared files, that are executed only once for each activation of a muj program.

Individual user housekeeping or end-of-session processing is coded in the terminal session section. For example, a muj program could be structured to maintain and display counts of transactions processed for individual users. The code to initialize the data areas for the counts would be placed in the beginning of the section, or in an area that the user executes the first time through the section, before terminal input/output processing begins. The code to display the counts would be placed in a segment of the terminal session section that is branched to when a user is ready to leave the program.

Individual user housekeeping and end-of-session processing are possible because of the way the program can be structured. The path the user takes while processing the terminal session section of the program is determined by the input from the user's terminal. By placing the initialization code ahead of the code for terminal input/output, the user can avoid branching back to the initialization code. When the user is ready to leave the program and inputs the data that signals this request, the program can branch out to do individual end-of-session processing, such as displaying the counts maintained for that user, before disconnecting the user from the program.

WRAP-UP PROCESSING

The wrap-up section is the last section of a COBOL muj program; SCED causes the wrap-up section to be entered after the last user executing in the program is disconnected. The code for wrap-up, or a branch to the code, must follow the call to disconnect users from the program. End-of-job processing, such as closing and detaching files and displaying job counts, is coded in the wrap-up section of the program.

The wrap-up code, similar to the initialization code, is executed only once for each activation of a muj program. This code is ignored when a user is disconnected while other users are still in execution in the program, and execution continues for active users.

ENTERING AND EXITING A COBOL MUJ PROGRAM

The sections of a COBOL muj program that are executed for each terminal user when three terminal users request processing of the program simultaneously are shown in figure 2-1. Terminal user A, the first user acknowledged by the system, causes the program to begin executing at the beginning of the initialization section, #1. Terminal users B and C, whose requests are acknowledged after the program is in execution, connect to the program at the beginning of the terminal session section, #2.

In the figure the users leave the program in the order in which they entered the program; however, the order in which a user leaves the program depends on several factors, such as the amount of activity the user has and the paths the user executes within the terminal session section. Terminal users A and B leave the program at the end of the terminal session section, #3; terminal user C, the last user to leave the program, causes the wrap-up code to be executed and, in effect, leaves the program at #4. (All users are connected and disconnected at the same points in the program, #2 and #3 respectively.)

DATA AREA ORGANIZATION

Special organization of some data areas is required in the Data Division of the COBOL muj program to ensure that these areas can be accessed as desired by several users. One of the problems that can arise if data is not organized correctly in the Data Division of a muj program is described in the following example: two users are waiting at the same SCED point for a terminal input operation. The statement at that point requests SCED to read data from the terminal into data area A. Both users will read data into the same area. The second user's data could overwrite the first user's data unless the data is protected in some way.

Certain data areas associated with one user must be kept independent of the data areas associated with other users; other data areas can be shared, either freely or under SCED control, among users. With this in mind, all data areas can be classified as either unprotected data areas or protected data areas.

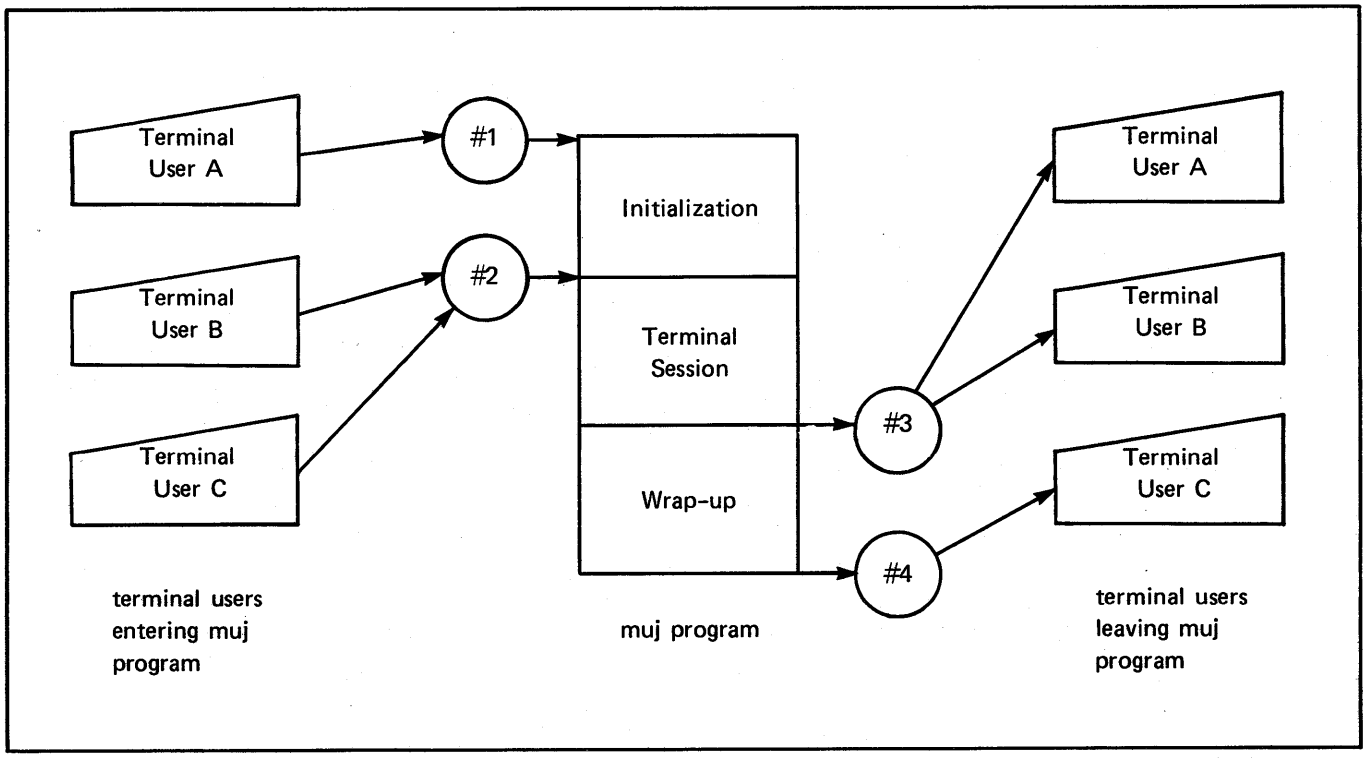

Figure 2-1. COBOL Muj Program Execution

UNPROTECTED DATA AREAS

Unprotected data areas are those data areas that can be referenced by all users as needed. Unprotected data areas can be classified as constants, common data areas, and local data areas.

Constants

Constants are data areas that remain unchanged throughout program execution. These areas are initialized to a value during program load and can be referenced freely by users executing in any part of the program.

Common Data Areas

Common data areas are areas that do not belong to a single user. These areas are intended either for functions common to all users, such as maintaining a count of all transactions processed by the muj program, or as a record area for an output transaction file of all transactions processed by the program. Common data areas are usually rare in a muj program.

Local Data Areas

Local data areas are data areas that are utilized only in a path between SCED points, where no user conflicts can occur. These data areas are considered to be local to that path and do not need protection from destruction by other users. For example, a record is both read into an area from a file or terminal and processed within that area before the next SCED point is reached. No other user can overwrite the area because the first user does not relinquish control of the program. It is important to remember that the only time SCED can change execution from one user to another is when the user who is executing reaches a SCED point.

PROTECTED DATA AREAS

Protected data areas are data areas that allow individual users full control of the contents over a number of SCED points. Protection of the data areas is accomplished in two ways, both under SCED control: by assigning each user a private user area, or by allowing the user to control a data area for a certain period of time. The two types of protected data areas that result are called user areas, and common data areas protected by interlock, respectively.

User Area

The user area is a data area whose data is protected across SCED points. The muj subroutines maintain a separate copy of this data area for each user connected to a muj program. SCED copies the user area of an individual user into the defined user area of the program whenever that user is scheduled for executing; thus, data in the user area can be considered universally protected from overwrite by all users except the one assigned to the copy of that user area.

The user area is defined in the Working-Storage Section of the program. All the data in working storage can be defined within the user area of a muj program; however the user area should be kept as small as possible for maximum efficiency. Terminal output buffers and other data areas that require a separate copy to be maintained for each user must be defined within the user area.

Terminal output buffers must be defined within the user area because the transfer of terminal output might not be completed for one user before another user is scheduled for processing. When the SCED call for terminal output is issued, the content of the user's terminal output buffer is copied to a SCED buffer, from which it is transmitted to the user's terminal. If all the SCED buffers are full, however, the user's terminal output buffer cannot be copied to the SCED buffer until space is available. Since a second user can be scheduled for processing in the same part of the program during the time the first user is waiting for a SCED buffer, the second user can overwrite the terminal output area of the first user if it is not protected. Placing the terminal output area in the user area provides this protection because SCED saves the first user's copy of the user area before scheduling the second user for processing. Whenever a SCED buffer is available, SCED can copy the first user's terminal output area from the saved user area to the SCED buffer.

The following guidelines should be considered when coding the user area:

The terminal output buffers must be defined within the user area.

The terminal input buffers do not have to be defined within the user area because terminal input is not written to the specified area from the SCED buffer until immediately before the user corresponding to the input is scheduled for processing. If the content of a terminal input buffer is to be saved across SCED points, either the terminal input buffer can be defined in the user area, or the data can be moved to a save area defined in the user area.

Data cannot be preset in the user area. For example, the VALUE IS clause is prohibited.

Data areas that are referenced by CYBER Record Manager while a user is at a SCED point, such as record and key areas, cannot be defined in the user area. Since these data areas must be defined in specific sections of the program, they require another type of protection that is provided by interlocks.

Common Data Areas Protected by Interlocks

Common data areas protected by interlocks are data areas that do not belong to a particular user; any user can gain exclusive access to these areas for a period of time. Protection for such areas is provided through the assignment of an interlock, a SCED feature designed to ensure that only one user executes in a specified part of the program at one time. An interlock does not protect the data areas directly; protection is accomplished by disallowing access to the instructions that reference the data areas to all users except the user assigning the interlock.

Record areas of files accessed randomly in a COBOL muj program must be protected by interlocks because each user needs to ensure that these areas remain unchanged during record processing. The data areas cannot be protected by definition in the user area because they are referenced by CYBER Record Manager and must be defined in specific sections of the Data Division.

Some of the uses of common data areas protected by interlock are:

To protect mass storage file record areas.

To protect records that are to be used for on-line file updating.

To protect areas into which overlays are to be loaded.

INTERLOCKS

Interlocks are defined in the Working-Storage Section of a COBOL muj program as 77-level elementary items, with unique data names and values. The values assigned to the interlocks must not exceed the maximum number of interlocks specified at installation time minus one. For example, if the installation parameter allows for five interlocks within the muj program, the interlocks defined and referenced in the muj program must have a value 0 through 4.

Interlocks are reserved and released during program execution. All interlocks reserved by a user must be released by that user before the user can disconnect from the program.

A unique interlock should be associated with each random file. Each time a random file is to be accessed, the interlock associated with that file must be reserved. When record processing is complete, the interlock must be released.

All references to a data area protected by an interlock must fall within the range of the interlock. The range of the interlock includes all parts of the program that lie between the statements reserving and releasing the interlock. The range of the interlock is not necessarily a contiguous part of the program; it can be a complex network of code entered and left at a number of different SCED points.

For example, a file can be accessed many times in a program for different functions such as adding records, deleting records, or updating existing records. The code to accomplish each of these functions can be in different parts of the program. Before the code for each function is executed, the interlock assigned to that file must be reserved; the interlock must be released by the

user reserving the interlock before another user can execute the code for any of the functions for that file. The code for all references to that file is therefore protected by the interlock, or falls within the range of the interlock.

TERMINAL INPUT/OUTPUT BUFFERS

Transmission of data to and from the terminal occurs through SCED buffers; SCED, in turn, copies data to and from areas defined in the COBOL muj program after a terminal input/output operation is initiated.

For each terminal output operation, the COBOL muj program must indicate the size of the SCED buffer required for data transmission by attaching a buffer identification number to the output buffer. The buffer identification number is a unique number, beginning with zero, assigned to each different SCED output buffer size during installation to distinguish one buffer size from another. The buffer identification number is assigned to the buffers in the order in which the buffers are specified in the installation deck. For example, the first buffer specified in the installation deck is assigned the buffer identification number of 0.

SCED uses the buffer identification number to determine which SCED buffer to use for copying the output buffer from the program. The programmer must ensure that the buffer type identification number attached to the output buffer defines a SCED buffer large enough to accommodate the output buffer that is to be transmitted to the terminal.

Output buffers can be contained either in one group item or in different group items within the user area.

Input buffers do not have a buffer type identification number; however, the size of the input buffer must be indicated within the terminal input area. Input buffers specified within the program receive terminal input from SCED buffers. SCED truncates data received from a user terminal that cannot be accommodated in the input buffer within the program.

INPUT/OUTPUT FILES

Only files that reside on nonallocatable devices can be accessed by a COBOL muj program that interfaces with SCED; these files can be thought of as being common to all users. Most files used in a muj program are accessed randomly from many different terminals. Random file access in a muj program is practical because a SCED function can be entered immediately after file access is initiated with the SEEK verb, and the time required for file access is utilized in program sharing.

The following procedure should be used for random access of indexed sequential and direct access files:

> Initiate file access with the SEEK verb.
>
> Issue a call to the SCED function IOWAIT to relinquish program control while the file record is being located by the SEEK operation.

Using the above procedure for accessing indexed sequential and direct access files allows SCED to share the file access time for one user with processing time for other users. The sharing is possible because the SEEK verb allows access time and processing time to overlap.

It is sometimes desirable to process a sequential file in a COBOL muj program. Processing of a sequential file cannot, and need not, involve a SCED function. There is no need to make the access of a sequential file into a SCED function because no significant time is involved in file manipulation; therefore, there is no advantage to sharing the access time involved.

A situation where it might be desirable to use a sequential file in a muj program is one in which all transactions input from the terminals are logged. The transactions are written to the log file in the order in which they are processed by the program. The disk operation of the sequential WRITE function can easily keep up with the relatively slow terminal input operation, which limits the overall execution speed of the program.

An individual user processing in a COBOL muj program built with SCED cannot create or attach any permanent or local files. All file manipulation must be handled within the program itself. For example, if a program is to access a permanent file, the file must be attached during program initialization, prior to opening the file, and detached during program wrap-up, after closing the file.

COBOL MUJ PROGRAM FLOWCHART

Figure 2-2 shows a COBOL muj program flowchart that includes most of the principles of writing a COBOL muj program discussed in section 2. The three parts of the program – initialization, terminal session, and wrap-up – are identified, along with the types of functions that are performed in each part.

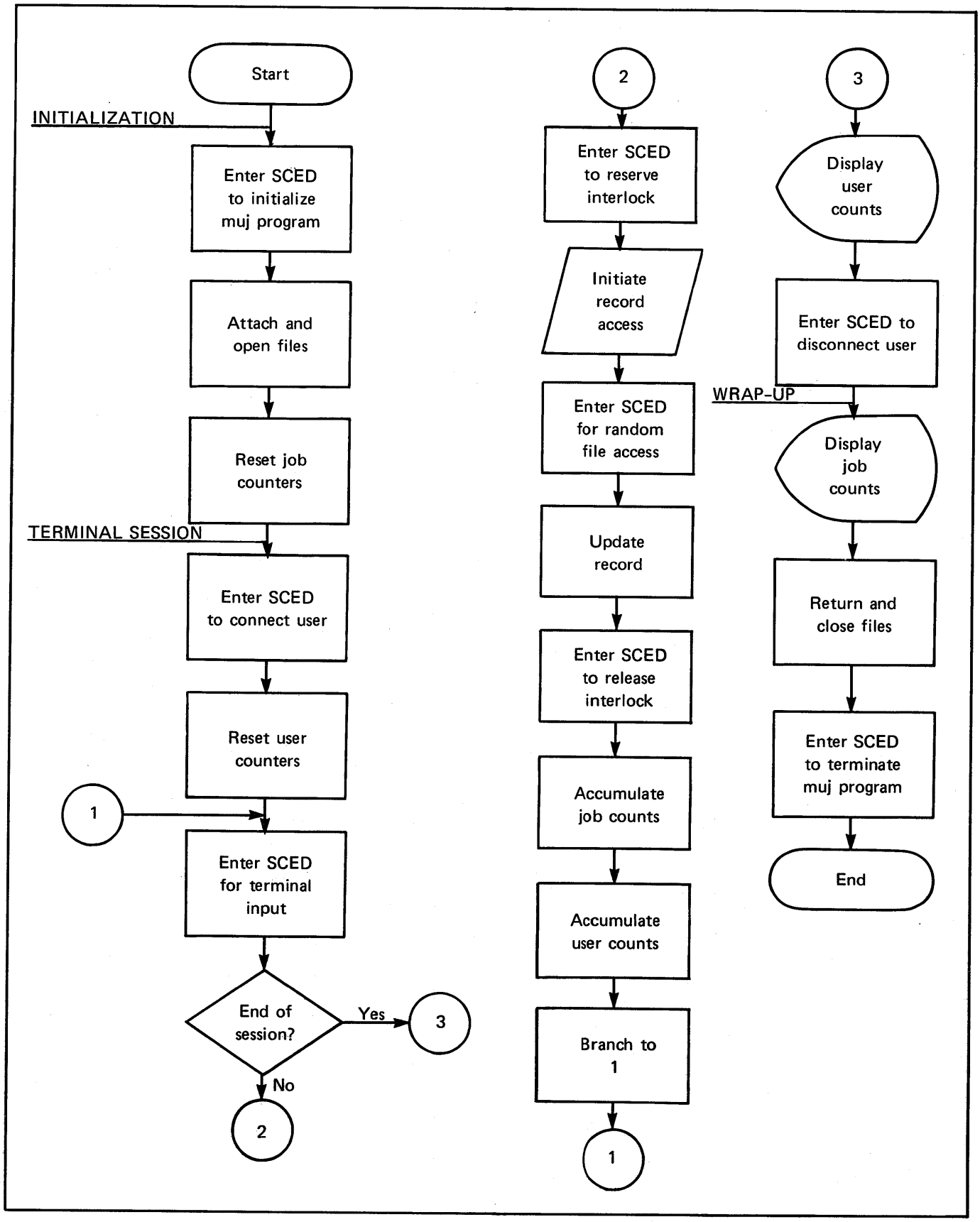

Figure 2-2. COBOL Muj Program Flowchart

SPECIFYING SCED CALLS

The SCED calls used to write a COBOL muj program are presented by the sections in which they are called, as indicated below.

INITIALIZATION
- INIT

TERMINAL SESSION
- CONNECT
- GETINT
- RETINT
- IOWAIT
- TERMIN
- TERMOUT
- DISCON

WRAP-UP
- EXITMUJ

The discussions of GETINT and RETINT are combined because of the interaction of these two calls.

Parameters specified on the SCED calls are required and must be in the order shown.

INIT

ENTER INIT USING user-area, paragraph-name$_1$, paragraph-name$_2$.

user-area	The data-name of the user area. SCED ensures that the appropriate user area is copied into the program for each user scheduled for program execution.
paragraph-name$_1$	The paragraph name of the initialization section.
paragraph-name$_2$	The paragraph name of the routine that is to be executed if the user is abnormally disconnected at any time during processing. This can be the same routine the user executes to disconnect from the muj program when terminal processing is completed.

The INIT call initializes the program for execution and sets up the linkages to disconnect code. The call specifies the location of a number of parts of a muj program and must be issued somewhere in the initialization section, before the call to connect users.

A fatal muj error occurs during program execution if INIT is not the first SCED call executed by the program.

Figure 3-1 shows an example of the use of the INIT call. USER-AREA is the name of the user area; STARTAD is the name of the first paragraph of the initialization section; and DISCNT is the name of the paragraph that is to be executed for abnormal disconnects from the program.

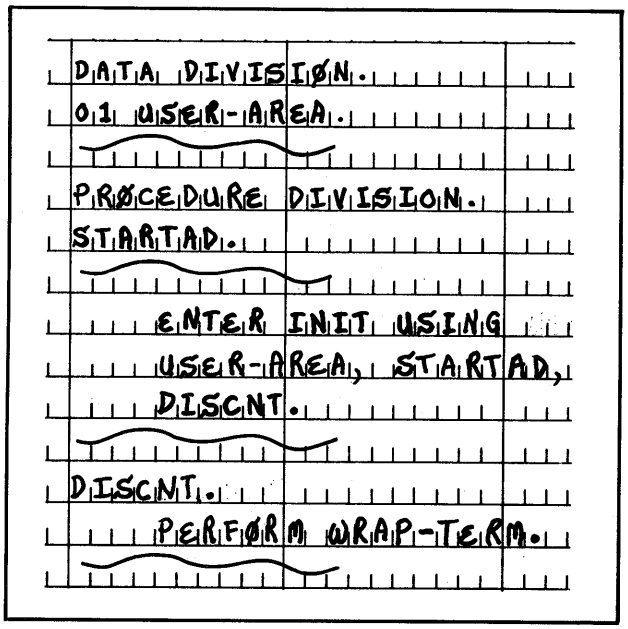

Figure 3-1. INIT Call Use

CONNECT

ENTER CONNECT.

This call connects a user to a muj program and is the first statement in the terminal session section. All users connecting to a running muj program begin execution at the ENTER CONNECT statement. When the statement is executed, a copy of the user area is created for the user, a user terminal identification number is generated and added to the table that SCED maintains for tracking users, and the user is scheduled for program execution.

Figure 3-2 shows an example of the use of the CONNECT call. When the statement ENTER CONNECT is executed, a user is connected to the muj program.

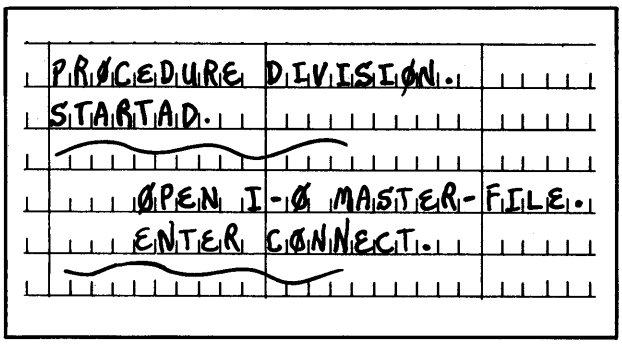

Figure 3-2. CONNECT Call Use

GETINT AND RETINT

ENTER GETINT USING interlock-id.

ENTER RETINT USING interlock-id.

interlock-id — The name of an independent item whose value identifies an interlock. Each interlock must be a 77 level, COMP-1 integer that is initialized to a unique value.

The call to GETINT reserves the named interlock. When the call is executed, one of the following actions occurs:

If the interlock is not already reserved, the user reserves the interlock and all other users are denied access to the instructions that fall within the range of the interlock.

If the interlock is already reserved, the user cannot advance beyond this statement until the interlock is released.

Program control does not return to the user requesting the reservation of an interlock until the interlock is reserved for that user.

The call to RETINT releases the named interlock. If the interlock specified in a call to RETINT was not previously reserved by the user requesting the release of the interlock, a fatal muj program error occurs.

The interlock values should be assigned consecutively, beginning with zero. If the program has ten interlocks, the values should range from 0 through 9.

Figure 3-3 shows an example of the use of the GETINT and the RETINT calls. The common data area FILE-REC is protected by an interlock. The statement ENTER GETINT reserves interlock INT-4, and the statement ENTER RETINT releases INT-4. Once a user executes the statement ENTER GETINT reserving the interlock, no other user can advance beyond this statement until the first user releases the interlock INT-4 by executing the statement ENTER RETINT.

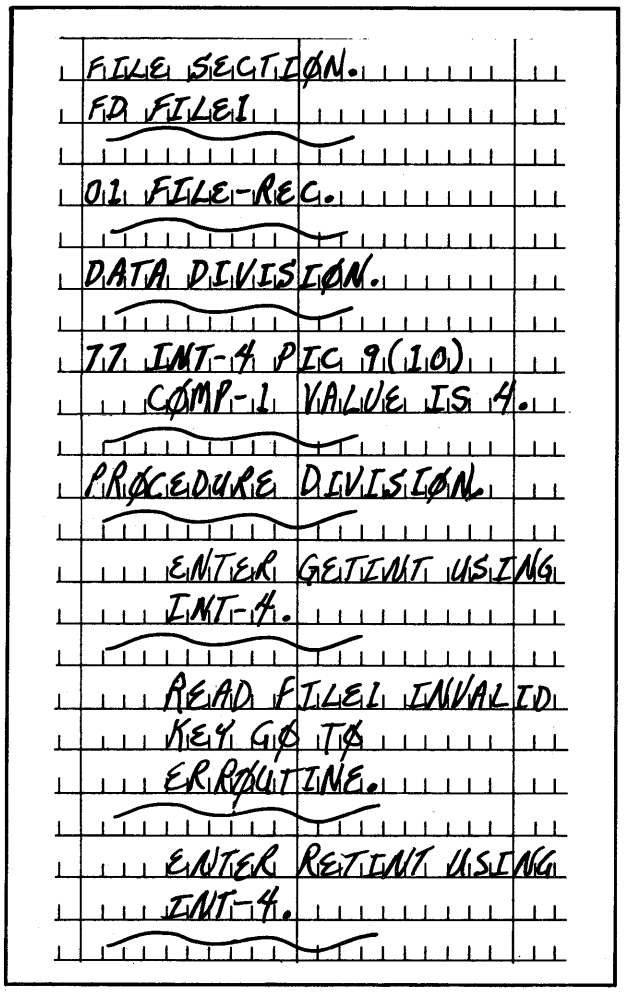

Figure 3-3. GETINT Call and RETINT Call Use

IOWAIT

ENTER IOWAIT USING file-name.

file-name — The name of a file for which an input/output operation was initiated with the SEEK verb.

The call to IOWAIT causes the user in execution to relinquish program control while a SEEK operation is performed on the specified file. Control does not return to the user until the SEEK operation is completed. This call must always fall within the range of an interlock.

Figure 3-4 shows an example of the use of the IOWAIT call. When the statement ENTER IOWAIT USING RANDOM-FILE is executed, the user executing the statement relinquishes program control while the SEEK operation is performed on RANDOM-FILE using KEY-1.

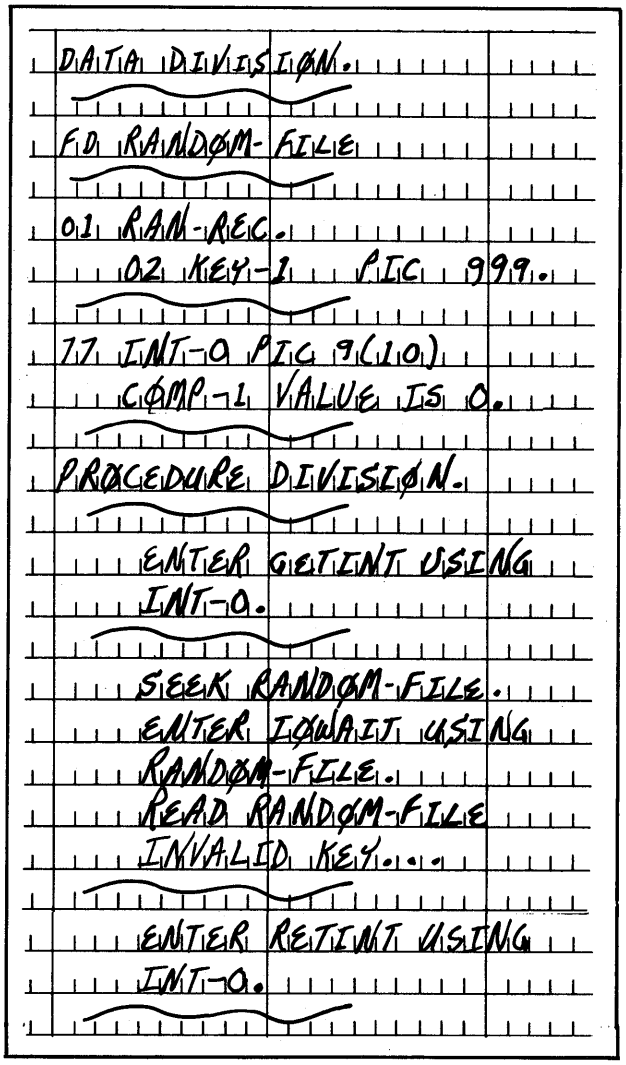

Figure 3-4. IOWAIT Call Use

TERMIN

ENTER TERMIN USING input-area.

input-area — The data name of the terminal input buffer area.

The call to TERMIN causes the user in execution to stop processing in the muj program until terminal input is received from the user's terminal. The input data is placed in the named input buffer area.

The first word of the terminal input buffer area must be a COMP-1 variable that is set to the size of the input buffer, in words, before the area is referenced in a SCED call. If the line of data received from the user's terminal is larger than the input buffer, the excess data is dropped.

Processing continues for the user at the statement following the call to TERMIN when the user enters at least one line of input at the terminal.

Figure 3-5 shows an example of the use of the TERMIN call. INPUT-AREA is the name of the terminal input buffer area, which is a common data area. INBUF-SIZE, the first word of the terminal input buffer area, is set to 9, which is the size of the input buffer INBUF. Setting INBUF-SIZE to 9 allows nine words of input data to be received at one time.

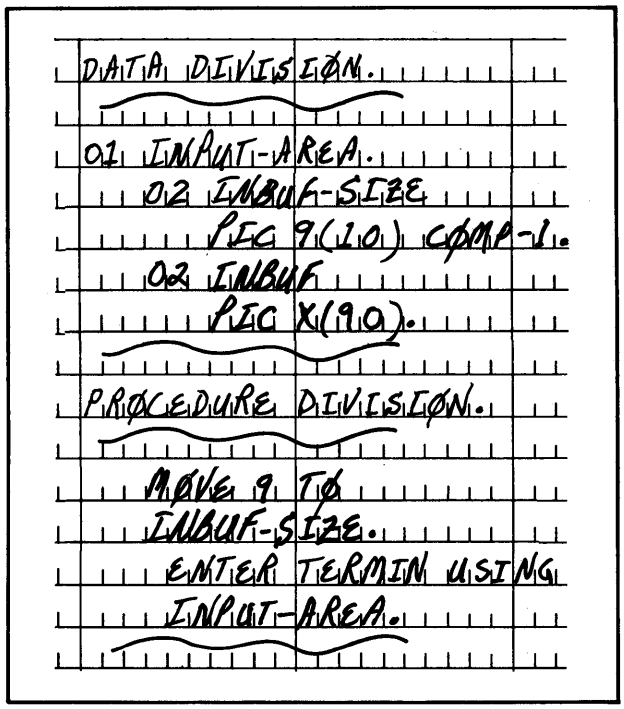

Figure 3-5. TERMIN Call Use

TERMOUT

ENTER TERMOUT USING output-area.

output-area	The data name of a terminal output buffer area that is defined in the user area.

The TERMOUT call transmits terminal output from the named terminal output buffer area. The terminal output buffer area must contain three elements: the buffer type, the buffer size, and the output buffer.

The buffer type, which is the first word of the terminal output buffer area, must be defined as a COMP-1 integer and set to the buffer type identification number before terminal output is performed. The buffer type identification number must specify a SCED buffer that is large enough to accommodate the output buffer area.

The buffer size, which is the second word of the terminal output buffer area, must be defined as a COMP-1 integer and set to the size of the output buffer in words before terminal output is performed.

The remainder of the terminal output buffer area, the output buffer, contains the data that is to be trasmitted to the terminal. Any number of lines can be sent to the terminal with one call to TERMOUT. Each line must begin with an INTERCOM carriage control character and terminate with a 12-bit byte of binary zeros in bits 0 through 11, as required by INTERCOM to define end-of-line.

The INTERCOM carriage control characters that can be specified are explained in the discussion of interactive output, below, and in table 3-1.

A muj program can be organized to transmit different output buffers from the same terminal output buffer area; however, each time a different type of output buffer is to be transmitted, the buffer type, the buffer size, and possibly the carriage control character must be changed to correspond to the output buffer that is to be transmitted to the terminal.

Fatal muj errors occur during terminal output operations if the terminal output area is not formatted correctly.

Interactive Output

Carriage control characters should be specified for all terminal output. The carriage control character used depends on whether the terminal is a Teletype or a display terminal. Table 3-1 illustrates the effect of carriage control characters before and after print action at Teletype or display terminals. The letters CR specify a Carriage return to beginning-of-line; the letters LF specify a line feed to the next line.

TABLE 3-1. INTERCOM 4 TELETYPE AND CRT CARRIAGE CONTROL CHARACTERS

Teletype†			CRT††	
Character	Before Output	After Output	Before Output	After Output
1	CR,3LF	None	CLEAR WRITE††	New line
*	CR,3LF	None	RESET WRITE	New line
+	CR	None	None	New line
0	CR,LF,CR,LF	None	New line	New line
-	CR,LF,CR,LF,CR,LF	None	New line, new line	New line
blank	CR,LF	None	None	New line

†INTERCOM returns an LF (line feed) after an input line from a Teletype; this effectively adds an additional LF for each of the carriage control characters for an output line that immediately follows an input line.

††Output to a CRT causes the cursor to be positioned at the beginning of the next line. CLEAR WRITE causes the screen to be cleared, and output starts at the top of the screen. RESET WRITE does not cause the screen to be cleared; output starts at the top of the screen and overwrites the existing display.

When the character Q or R is used for carriage control, no output takes place, the remainder of the line is ignored, and the following action is specified:

Q Clear auto page wait.

R Select auto page wait.

These actions apply to a display terminal only.

INTERCOM normally enters page wait at a display terminal if a full screen of information has been output since the last input was entered from the terminal, or if a CLEAR WRITE or RESET WRITE is followed immediately by an output to the terminal. The character Q causes INTERCOM to suspend an automatic page wait temporarily, until either the character R reselects automatic page wait, or the end of the program is reached.

Figure 3-6 shows an example of the use of the TERMOUT call. The terminal output buffer area OUTPUT-AREA is capable of transmitting up to 10 lines of data with one call to TERMOUT. Each data line is nine words long, with the last word containing a 12-bit byte of binary zeros in bits 0 through 11. Each different size of output buffer that is formatted and transmitted from this terminal output buffer area must have a unique buffer type.

The Procedure Division shows the instructions for formatting one output buffer. This output buffer is 18 words (two lines of data) long and has the buffer type identification number of zero.

The statement MOVE CHAR10 to CHAR1, or its equivalence, is required prior to executing the statement MOVE ALL-ZERO-BITS TO LINE-TERM (1). Execution of the MOVE statement ensures that the COMP-1 item CONST-ZERO is set to an unpacked integer (a 12-bit byte of binary zeros).

BUFFER-TYPE is set to zero, the buffer type identification number, to indicate the output buffer type that is to be transmitted.

BUFFER-SIZE is set to 18 to indicate the size of the output buffer.

CARR-CONT is set to zero. This causes a new line of output to be written to the CRT output screen without clearing the screen or overwriting existing data.

The first character of the second line in the output buffer is set to a minus to provide for triple spacing of the second line terminal output.

When the statement ENTER TERMOUT is executed, the first 18 words of the output buffer OUTBUF are transmitted to the user's terminal; this causes two lines of data to be displayed upon the user's screen.

```
DATA DIVISION.

01 OUTPUT-AREA.
   02 BUFFER-TYPE PIC 9(10) COMP-1.
   02 OUTBUF-SIZE PIC 9(10) COMP-1.
   02 OUTBUF      PIC X(900).
   02 LINE-BY-LINE REDEFINES OUTBUF.
      03 ONE-LINE OCCURS 10.
         04 CARR-CONT PIC X.
         04 DATA-LINE PIC X(79).
         04 LINE-TERM PIC X(10).

01 CONST-ZERO PIC 9(10) COMP-1 VALUE 0.
01 ALL-ZERO-BITS REDEFINES CONST-ZERO.
   02 CHAR1  PIC X.
   02 FILLER PIC X(8).
   02 CHAR10 PIC X.

PROCEDURE DIVISION.

   MOVE CHAR10 TO CHAR1.

   MOVE "0" TO CARR-CONT (1).
   MOVE "-" TO CARR-CONT (2).
   MOVE "LINE ONE" TO DATA-LINE (1).
   MOVE "LINE TWO" TO DATA-LINE (2).
   MOVE ALL-ZERO-BITS TO LINE-TERM (1), LINE-TERM (2).
   MOVE 0 TO BUFFER-TYPE.
   MOVE 18 TO OUTBUF-SIZE.
   ENTER TERMOUT USING OUTPUT-AREA.
```

Figure 3-6. TERMOUT Call Use

DISCON

ENTER DISCON.

This call disconnects a user from the muj program. Individual end-of-session processing must be completed before the call is executed.

When the call to DISCON is executed, one of the following actions occurs:

If the user executing the statement is the last user connected to the program and no interlocks are reserved for that user, the wrap-up code is executed.

If the user executing the statement has interlocks reserved, a fatal muj program error occurs.

If the user executing the statement is not the last user connected to the program and has no interlocks reserved, the user is disconnected from the program, the wrap-up code is ignored, and program execution continues for active users.

The wrap-up code must follow the call to DISCON and should be as brief as possible because the last user cannot execute any further INTERCOM or operating system commands until wrap-up is finished.

Figure 3-7 shows an example of the use of the DISCON call. A terminal user must enter END at the terminal to disconnect from the program.

EXITMUJ

ENTER EXITMUJ.

This call terminates a muj program. It replaces STOP RUN, the normal COBOL statement for program termination. The call to EXITMUJ must be the last statement in the wrap-up section of the program. The statement is executed after the last user processing the program is disconnected from the program.

```
DATA DIVISION.
01 INPUT-AREA.
   02 INBUF-SIZE
      PIC 9(10) COMP-1.
   02 INBUF.
      03 IN-TYPE
         PIC X(3).
      03 FILLER
         PIC X(87).
PROCEDURE DIVISION.

   ENTER TERMIN USING
   INPUT-AREA.
   IF IN-TYPE = 'END'
   GO TO DISCON.

DIS-CON.
   ENTER DISCON.
```

Figure 3-7. DISCON Call Use

Figure 3-8 shows an example of the use of the EXITMUJ call. After the statement ENTER EXITMUJ is executed, the program is terminated unless a new user requests the muj program during program wrap-up. If a new user requests the muj program during wrap-up, SCED returns control to the initialization section, STARTAD, and reinitializes the program.

PLACEMENT OF SCED CALLS

SCED calls should be placed in the COBOL muj program in a manner that ensures the fastest response time possible for all users, if efficient program sharing is to be a reality. The following guidelines should be used in writing the program:

1. INIT must be the first SCED call in a muj program; it appears only once.

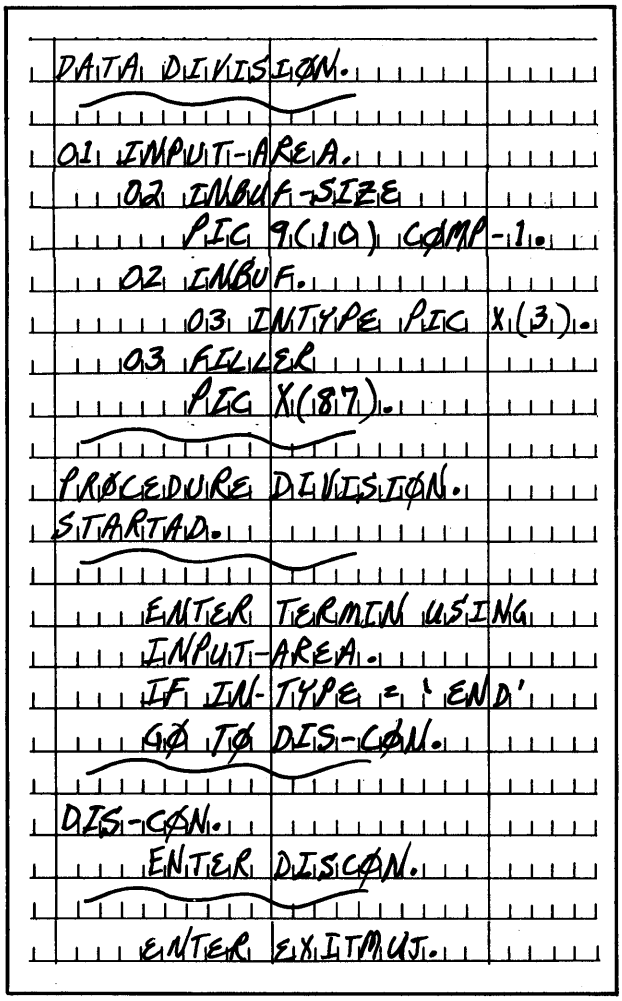

Figure 3-8. EXITMUJ Call Use

2. CONNECT must be the second SCED call in a muj program; it appears only once.

3. GETINT and RETINT can appear any number of times in the terminal session section of a muj program.

4. TERMIN and TERMOUT can appear any number of times in a muj program, but these calls should not be placed within the range of an interlock.

5. IOWAIT can appear any number of times in a muj program. The call to IOWAIT must be preceded by a SEEK operation and fall within the range of an interlock.

6. The call to DISCON must be the last statement executed by users in the terminal session section. If the call to DISCON is coded more than once in the muj program, it must be followed by the wrap-up code or a branch to the wrap-up code each time it appears.

7. The call to EXITMUJ must be the last statement in the wrap-up section.

Special care should be taken in the placement of TERMIN, TERMOUT, GETINT, and RETINT to keep response time to a minimum. Calls to TERMIN and TERMOUT should not be placed within the range of an interlock because terminal input and output operations are time-consuming, and improper placement of the calls can defeat the purpose of efficient program sharing. For example, if a call to TERMIN were placed within the range of an interlock and the terminal user was unable to respond to a request for input at once, that portion of the program that falls within the range of the interlock might be tied up for a long time. Other users would be denied access to the instructions that fall within the range of the interlock.

Figure 3-9 shows two segments of flowcharts that have identical functions charted. If a muj program were coded to include segment A, sections of the program could be tied up in execution for one user for indefinite time periods by the improper placement of the calls to TERMIN and TERMOUT. Segment B shows the proper placement of the SCED calls to TERMIN and TERMOUT.

For each call to GETINT executed to reserve an interlock in a muj program, a call to RETINT must be executed to release the interlock. This does not mean, however, that the program must contain the same number of calls to RETINT and GETINT. One call could be made to reserve an interlock in the program, but any number of calls could be issued to release the interlock; and the converse is true. For example, when different functions are possible for a file, the programmer can perform the following operations: the interlock can be set, the file record can be read into the record area of the program, and the program can branch to one of a number of routines to perform the required function. It might be desirable to issue a call to release the interlock in each routine where a file function is performed, rather than to branch to a common routine to release the interlock. If the program were coded in this manner, more calls would be coded to release the interlock than to reserve the interlock, but the number of executions of the calls to release and reserve the interlock would maintain a one-to-one ratio.

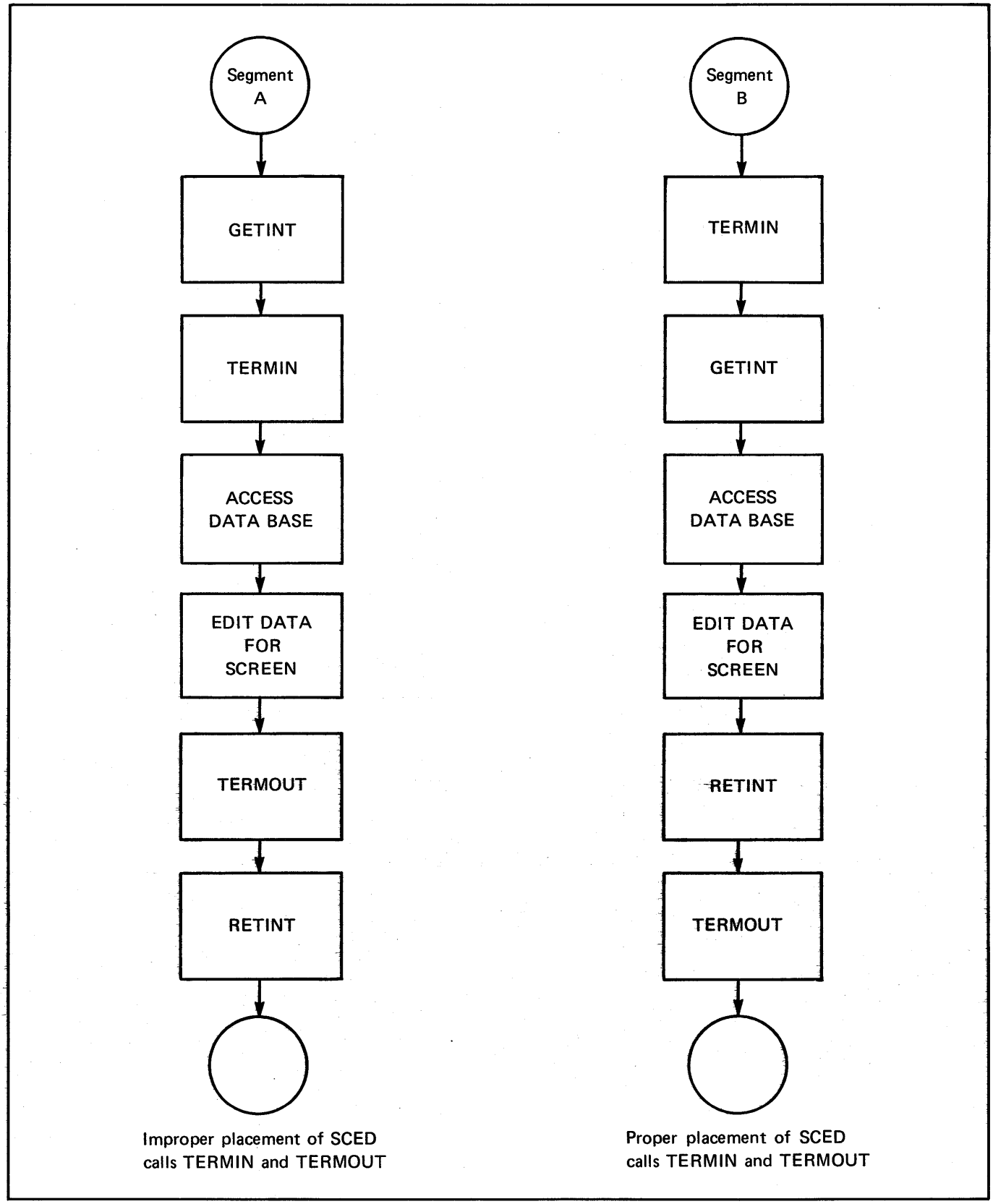

Figure 3-9. Placement of SCED Calls

PROCEDURE LINKAGE IN COBOL

COBOL procedures called by a PERFORM statement that can be executed from more than one path should not contain a SCED call because the procedure return information COBOL stores in central memory for one user can be destroyed by another user. For example, the first user's procedure return information would be destroyed if the following conditions exist:

- A user executes a procedure with a PERFORM statement and reaches a SCED point where SCED schedules other users for program execution
- Another user performs the procedure from a different path before control is returned to the first user

The COBOL reference manual contains a complete discussion of the execution of procedures by means of the PERFORM statement.

FILE ASSIGNMENT

Files referenced in a COBOL muj program must be attached and returned in the program by calls to subroutines coded in a language that can reference RA+1. Examples of COMPASS subroutines that can be used to attach and return permanent files are shown in figure 3-10. These subroutines are entered by including the statements ENTER ATTACH and ENTER DETACH in the COBOL muj program.

Execution of the ATTACH subroutine causes the permanent file, whose logical and permanent file name is IOFILE and whose permanent file ID is SCEDF1, to be attached to the muj program. ATTACH must be entered in the initialization section of the program before IOFILE is opened. IOFILE must also be named in a SELECT clause in the FILE-CONTROL paragraph of the muj program.

PROGRAM ATTACH – Subroutine to Attach Permanent File IOFILE

```
          IDENT   ATTACH
          ENTRY   ATTACH
ATTACH    EQ      *+1S17                      USED FOR DEBUGGING
          ATTACH  IOFILE,RC                   ISSUES AN ATTACH REQUEST
          EQ      ATTACH                      RETURNS TO CALLING PROGRAM
IOFILE    FDB     IOFILE,IOFILE,ID=SCEDF1     SPECIFIES PERMANENT FILE
                                                  PARAMETERS
          END
```

PROGRAM DETACH – Subroutine to Return Permanent File IOFILE

```
          IDENT   DETACH
          ENTRY   DETACH
DETACH    EQ      *+1S17                  USED FOR DEBUGGING
          SB1     =XIOFILE                GETS ADDRESS OF THE FIT
          SA1     B1+1                    FETCHES ADDRESS OF THE FET
          SA1     X1                      FETCHES FIRST WORD OF THE FET
          RJ      =XCPC                   CALLS CPC TO RETURN THE FILE
          VFD     18/7,1/0,1/1,40/174B    DEFINES VARIABLE FIELDS FOR
                                              CALL TO CPC
          EQ      DETACH                  RETURNS TO CALLING PROGRAM
          END
```

Figure 3-10. COMPASS Subroutines to Attach and Return a Permanent File

Execution of the subroutine DETACH causes the file named IOFILE to be detached from the system. DETACH must be entered in the wrap-up section of the program, after the file is closed.

Each permanent file referenced in the COBOL muj program must be attached and returned by calls to subroutines.

SAMPLE COBOL MUJ PROGRAM

The COBOL programmer should now have some idea of how to write a COBOL muj program that interfaces with SCED. The new programming concepts that are presented in this manual are (1) establishing SCED points, and (2) protecting data areas.

The COBOL muj program shown in figure 3-11 illustrates the incorporation of the concepts of establishing SCED points and protecting data areas for interface with SCED. The program updates a file from information entered by users at various terminals and accepts five different types of input transactions as listed below:

Type	Function
1	Add a record to the file.
2	Add the amount entered to an existing file record.
3	Subtract the amount entered from an existing file record.
4	Delete an existing record from the file.
E(ND)	Disconnect a user from the program.

The file that is updated during processing of the COBOL muj program contains 28-character records, which have the following fields:

ACCOUNT NUMBER	PIC 9(3).
FILLER	PIC 9(15).
BALANCE	PIC 9(10).

SCED installation parameters used to install the sample program are listed below:

SCED Installation Parameters	Specification
MAXUSR 10	The maximum number of users that can be connected to the program at one time is 10.
USAREA 2,15	Two user areas are to be allocated in central memory; each user area is 15 central memory words in length.
NUMINT 10	The maximum number of interlocks that can be referenced in the program is ten. Interlocks referenced in the program must be assigned a value in the range 0 through 9.
DEFBUF	Output buffers defined by OUTBUF are included in the program.
OUTBUF 4,12	An output buffer type that is 12 central memory words in length is specified. Four buffers of this type are allocated in central memory. The buffer type identification of this buffer is 0.

The sample program is compiled with the E option of the COBOL compiler, where E=PSRCN; consequently, PSRCN is the name entered at a terminal to call the muj program.

The discussion of the program is presented by divisions; a discussion of the IDENTIFICATION DIVISION is omitted.

ENVIRONMENT DIVISION

The SPECIAL-NAMES paragraph is included so that CONSOLE can be equated to the mnemonic name SCREEN, and output can be written to the dayfile by referencing SCREEN in a DISPLAY statement.

```
IDENTIFICATION DIVISION.
PROGRAM-ID.   MUSCED.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  6600.
OBJECT-COMPUTER.  6600.
SPECIAL-NAMES.
    CONSOLE  IS SCREEN.
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT MASTER-FILE ASSIGN TO IOFILE
        ORGANIZATION IS INDEXED
        ACCESS IS RANDOM
        SYMBOLIC KEY IS I-O-KEY
        INDEX-BLOCK CONTAINS 10230 CHARACTERS
        RECORD-BLOCK  CONTAINS 10230 CHARACTERS
        INDEX-PADDING IS 15 PERCENT
        DATA-PADDING IS 50 PERCENT.

DATA DIVISION.
FILE SECTION.
FD  MASTER-FILE
    LABEL RECORDS OMITTED
    DATA RECORD IS I-O-REC.
01  I-O-REC.
    02  ACCT          PIC 9(3).
    02  FILLER        PIC 9(15).
    02  BAL           PIC  9(10).


WORKING-STORAGE SECTION.
77  INT-1     PIC 9(10)  COMP-1  VALUE 1.
77  I-O-KEY   PIC 9(3) USAGE COMP.
77  IN-CNT    PIC 9(3).
77  DEL-CNT   PIC 9(3).
77  NEW-CNT   PIC 9(3).
77  SUB-1     PIC 99.
77  SUB-2     PIC 99.

01  MSG1  PIC  X(32)  VALUE ≠NUMBER OF TRANSACTION APPLIED    ≠.
01  MSG2  PIC  X(32)  VALUE ≠NUMBER OF RECORDS DELETED        ≠.
01  MSG3  PIC  X(32)  VALUE ≠NUMBER OF RECORDS ADDED          ≠.
```

Figure 3-11. Sample COBOL Muj Program (Sheet 1 of 5)

```
01   INPUT-AREA.
     02   INBSIZE      PIC 9(10) COMP-1.
     02   IN-BUF.
          03   IN-TYPE      PIC X.
          03   FILLER       PIC X.
          03   IN-ACCT      PIC X(3).
          03   FILLER       PIC X.
          03   IN-AMT-1     PIC X(10).
          03   IN-AMT-2 REDEFINES IN-AMT-1.
               04   IN-AMT OCCURS 10 TIMES PIC X.
          03   FILLER       PIC X(4).

01   USER-AREA.
     02   OUTPUT-AREA.
          03   OT-BUF-1     PIC 9(10) COMP-1.
          03   OT-BUF-S     PIC 9(10) COMP-1.
          03   OT-BUF.
               04   FORMAT-CHAR PIC X.
               04   OT-ACCT  PIC X(3).
               04   OT-MSG   PIC X(57).
               04   OT-BAL   PIC X(10).
               04   FILLER   PIC X(30).
          03   OT-BUF-END  PIC 9(10) COMP-1 VALUE ZERO.
          03   OT-ZERO   REDEFINES OT-BUF-END.
               04   CHAR-ZE  PIC X.
               04   FILLER   PIC X(8).
               04   MOVE-ZE  PIC X.
     02   SAVE-AREA.
          03   S-ACCT       PIC 9(3).
          03   S-TYPE       PIC 9.
          03   S-AMT        PIC 9(10).
          03   S-AMT-2 REDEFINES S-AMT.
               04   S-AMT-1 OCCURS 10 TIMES PIC 9.
          03   S-BAL        PIC  9(10).

PROCEDURE DIVISION.
INITIALIZE.
     MOVE ZEROS TO IN-CNT, DEL-CNT, NEW-CNT.
     MOVE 2 TO INBSIZE.

     ENTER INIT USING USER-AREA, INITIALIZE, DIS-CON.
     ENTER ATTACH.
     OPEN I-O MASTER-FILE.

     ENTER CONNECT.
```

Figure 3-11. Sample COBOL Muj Program (Sheet 2 of 5)

```
    MOVE MOVE-ZE TO CHAR-ZE.
    MOVE 12 TO OT-BUF-S.
    MOVE ZEROS TO OT-BUF-T, FORMAT-CHAR.
    MOVE SPACES TO OT-BUF.
    MOVE ≠MUSCED PROGRAM READY.≠ TO OT-BUF.

IN-PROC-1.
    ENTER TERMOUT USING OUTPUT-AREA.
    MOVE SPACES TO OT-BUF.

    MOVE ≠ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR
    ≠END TO TERMINATE≠ TO OT-BUF.
    ENTER TERMOUT USING OUTPUT-AREA.
    MOVE SPACES TO OT-BUF.
    MOVE SPACES TO IN-BUF.
    MOVE ZEROS TO S-AMT.

    ENTER TERMIN USING INPUT-AREA.

    IF  IN-TYPE = ≠E≠
        GO TO DIS-CON.
    IF IN-TYPE = ≠1≠
        OR IN-TYPE = ≠4≠
        GO TO BY-PASS.
    IF IN-TYPE = ≠2≠
        OR IN-TYPE = ≠3≠
        GO TO FORMAT  ELSE
        MOVE ≠INVALID TYPE CODE≠ TO OT-BUF.
        GO TO IN-PROC-1.

FORMAT.
    MOVE 1 TO SUB-1.
LOOP-1.
    IF IN-AMT (SUB-1) = ≠,≠ GO TO LOOP-2.
    ADD 1 TO SUB-1.
    IF SUB-1 LESS THAN 12 GO TO LOOP-1.
    MOVE ≠INVALID AMT. - TYPE 2 OR 3 TRANSACTION≠ TO OT-MSG.
    GO TO IN-PROC-1.
LOOP-2.
    IF SUB-1 LESS THAN 2
        GO TO BY-PASS.
    MOVE 10 TO SUB-2.
LOOP-3.
    SUBTRACT 1 FROM SUB-1.
    MOVE IN-AMT (SUB-1) TO S-AMT-1 (SUB-2).
    IF SUB-1 = 1 GO TO BY-PASS.
    SUBTRACT 1 FROM SUB-2.
    GO TO LOOP-3.
```

Figure 3-11. Sample COBOL Muj Program (Sheet 3 of 5)

```
BY-PASS.
     MOVE IN-TYPE TO S-TYPE.
     MOVE IN-ACCT TO S-ACCT.

     ENTER GETINT USING INT-1.

     MOVE S-ACCT TO I-O-KEY.
     SEEK MASTER-FILE RECORD.

     ENTER IOWAIT USING MASTER-FILE.

     READ MASTER-FILE INVALID KEY GO TO TYPE-1.
     IF S-TYPE = 4 GO TO DELETE-REC.
     IF S-TYPE = 2 ADD S-AMT TO BAL.
     IF S-TYPE = 3 SUBTRACT S-AMT FROM BAL.
     REWRITE I-O-REC INVALID KEY
          MOVE ≠UNABLE TO WRITE MASTER FILE RECORD.    ≠
          TO OT-BUF
          GO TO RELEASE-INTERLOCK.
     ADD 1 TO IN-CNT.

UPDATE-COMPLETE.


     MOVE BAL TO S-BAL.
     MOVE S-ACCT TO OT-ACCT.
     MOVE ≠ - TRANSACTION APPLIED. BALANCE IS≠ TO OT-MSG.
     MOVE S-BAL TO OT-BAL.

RELEASE-INTERLOCK.
     ENTER RETINT USING INT-1.
     GO TO IN-PROC-1.

TYPE-1.
     IF S-TYPE = 1 NEXT SENTENCE
          ELSE MOVE S-ACCT TO OT-ACCT
          MOVE ≠ - INVALID ACCOUNT NO.  ≠ TO OT-MSG
          MOVE SPACES TO OT-BAL
          GO TO RELEASE-INTERLOCK.
     MOVE ZERO TO I-O-REC.
     MOVE S-ACCT TO ACCT.
     MOVE S-AMT TO BAL.
     WRITE I-O-REC INVALID KEY
          MOVE ≠UNABLE TO WRITE RECORD.          ≠ TO
          OT-BUF
          GO TO RELEASE-INTERLOCK.
     ADD 1 TO NEW-CNT.
     GO TO UPDATE-COMPLETE.
```

Figure 3-11. Sample COBOL Muj Program (Sheet 4 of 5)

```
    DELETE-REC.
        MOVE BAL TO S-BAL.
        DELETE RECORD FROM MASTER-FILE INVALID KEY
            MOVE ≠UNABLE TO DELETE RECORD.          ≠ TO OT-BUF
            GO TO RELEASE-INTERLOCK.
        ADD 1 TO DEL-CNT.
        MOVE S-ACCT TO OT-ACCT.
        MOVE ≠ - DELETED FROM FILE.  BALANCE WAS≠ TO OT-MSG.
        MOVE S-BAL TO OT-BAL.
        GO TO RELEASE-INTERLOCK.

    DIS-CON.
        ENTER DISCON.
        DISPLAY MSG1  IN-CNT UPON SCREEN.
        DISPLAY MSG2 DEL-CNT UPON SCREEN.
        DISPLAY MSG3 NEW-CNT UPON SCREEN.
        CLOSE MASTER-FILE.
        ENTER DETACH.

        ENTER EXITMUJ.
    7/8/9

        IDENT  ATTACH
        ENTRY ATTACH
ATTACH  EQ     *+1S17              ENTRY/EXIT...
        ATTACH IOFILE,RC
        EQ     ATTACH
IOFILE  FDB    IOFILE,IOFILE,ID=SCEDF1
        END
        IDENT  DETACH
        ENTRY DETACH
RETURN  EQ     *+1S17                  ENTRY/EXIT..
        SB1    =XIOFILE                ADDRESS OF THE FIT
        SA1    B1+1                    FETCH ADDRESS OF THE FET
        SA1    X1                      FETCH FIRST WORD OF THE FET
        RJ     =XCPC
        VFD    18/7,1/0,1/1,40/174B
        EQ     DETACH                  RETURN..
        END
```

Figure 3-11. Sample COBOL Muj Program (Sheet 5 of 5)

DATA DIVISION

MASTER-FILE is an indexed sequential file that is accessed randomly with the symbolic key I-O-KEY. The record description entry I-O-REC, a common data area, is protected by an interlock.

INT-1 is a constant that is used as the interlock that disallows all but one user access to the data in I-O-REC for a period of time.

IN-CNT, DEL-CNT, and NEW-CNT are common data areas used to maintain counts of records processed by the program.

SUB-1 and SUB-2 are local data areas used as subscripts in the program.

MSG1, MSG2, and MSG3 are constants that identify counts of records processed in the program. These counts are displayed during program wrap-up.

INPUT-AREA is a terminal input buffer area. In this program, the terminal input buffer is a common data area.

USER-AREA is the user area, a copy of which is maintained by SCED and the muj subroutine for each user connected to the program. OUTPUT-AREA is the terminal output area, and SAVE-AREA is a hold area for input transactions. Data is moved into SAVE-AREA from the input buffer IN-BUF, with the amount field being reformatted in the process, so that the data can be saved across SCED points.

PROCEDURE DIVISION

INITIALIZE, the first paragraph in the program, begins with the initialization code. The first SCED call, ENTER INIT, specifies that the user area is USER-AREA; the paragraph that is to be executed for program restart is INITIALIZE; and the paragraph that is to be executed if a user is disconnected abnormally from the program is DIS-CON. The subroutines to attach the file and the OPEN statement are executed in the initialization section because MASTER-FILE must be opened only once for each activation of the program.

The SCED call ENTER CONNECT signals the beginning of the sharable terminal session section. Users connecting to a running program begin executing at this instruction.

The five statements following ENTER CONNECT are executed only once for each user connecting to the program. Since only one output buffer is formatted and transmitted from the terminal output buffer area OUTPUT-AREA, the buffer size, buffer type identification number, and carriage control character can be set in the terminal output buffer area once for each user connecting to the program. These five statements can be regarded as individual user housekeeping code.

IN-PROC-1, the second paragraph, begins the loop of code that is executed for each transaction input by users. Two messages are displayed each time the loop is entered: the first message tells the user the processing just completed for the last transaction, if any, and the second message instructs the user to enter a transaction or the signal to disconnect from the program. The first time a user executes IN-PROC-1, the messages shown in figure 3-12 are transmitted to the user's terminal.

```
MJSCED PROGRAM READY

ENTER NEXT UPDATE IN FORMAT
TYPE,ACCOUNT,AMOUNT, OR END
TO TERMINATE
```

Figure 3-12. Terminal Messages from Program Example PSRCN

In each subsequent execution of the loop, the first message is replaced by a message that shows the status of the last transaction processed for the user. The second message is always the same as the second message shown above.

The program must now be readied for terminal input, and the SCED call ENTER TERMIN is executed to receive the data that is input by the user.

Once the terminal input operation is completed, the transaction type code IN-TYPE is checked, with the following results:

> If the code is E(ND), the program branches to DIS-CON, the name of the paragraph that is executed to disconnect users from the program.
>
> If the code is 1 or 4, the paragraph that reformats the amount field, FORMAT, is bypassed because type 1 and 4 transactions do not have amount fields.

If the code is invalid, the appropriate message is formatted, and the program branches to the beginning of the loop, IN-PROC-1.

If the code is 2 or 3, the code that reformats the amount field is executed. This code begins at paragraph FORMAT.

FORMAT is the first paragraph of the code that reformats the amount field. The amount field is reformatted because the amount can be entered by the user without leading zeros. If the amount field is blank or invalid, a message is formatted and the program branches to IN-PROC-1.

BYPASS is the paragraph that accesses the file for updating. The remainder of the input data that is to be saved across SCED points is moved to SAVE-AREA, which is in the user area. The interlock is then reserved, and the file is accessed on the symbolic key I-O-KEY that is set to the account number of the transaction.

The amount fields are applied when a type 2 or 3 transaction account number matches the account number of a record on the file. The program branches to the appropriate routines to delete and add records for type 1 and 4 transactions.

For all transactions, the record is counted, the output message indicating the status of the transaction is formatted, and the program advances to the routine to release the interlock, RELEASE-INTERLOCK.

TYPE-1 is executed when a transaction whose account number does not match an existing file record is entered by the user. A type 1 transaction causes a new record to be added to the file. After reformatting the appropriate message, the program advances to the section of the program that released the interlock.

DELETE-REC is executed when a type 4 transaction whose account number matches an existing file record is entered. A type 4 transaction causes the corresponding record to be deleted from the file. A message is formatted and the program branches to the routine to release the interlock.

The paragraphs TYPE-1 and DELETE-REC both fall within the range of the interlock. The program is structured so that the interlock is reserved and released once in establishing the range of the interlock; however, the program could have been written so that the interlock was released within these paragraphs and the subsequent branches from these paragraphs were made directly to IN-PROC-1.

DIS-CON is executed when a user is disconnected abnormally or enters END at a terminal. The user branching to DIS-CON executes the first statement if other users are connected to the program. The first statement in DIS-CON, ENTER DISCON, signals the end of the terminal session section and is followed by the wrap-up code.

When the last user executing in the program branches to this paragraph, the wrap-up code is executed. Wrap-up includes displaying counts and returning and closing MASTER-FILE. The final statement ENTER EXITMUJ terminates the program.

ATTACH, the COMPASS subroutine that attaches MASTER-FILE, is entered in the initialization section prior to opening the file.

DETACH, the COMPASS subroutine that returns the file, is entered in the wrap-up section of the program after closing the file.

Figure 3-13 shows a portion of IOFILE, the file that is updated during processing of the sample program shown in figure 3-11.

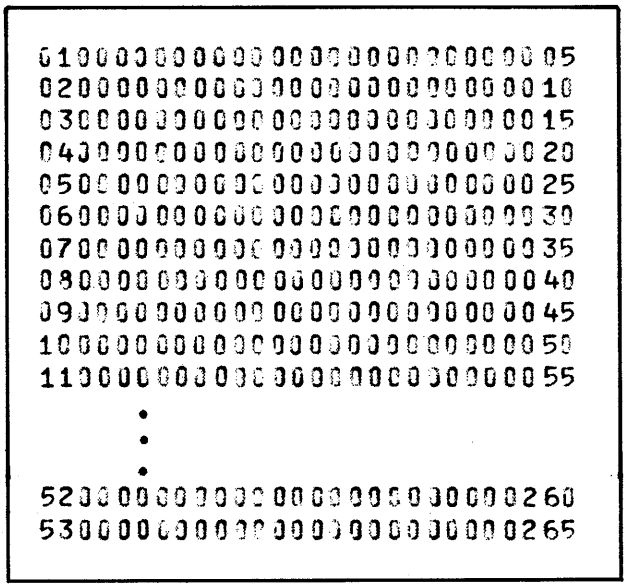

Figure 3-13. Partial Listing of IOFILE

Figure 3-14 shows a copy of the Teletype output that logs terminal input/output activity for a user executing the COBOL muj program. All activity, including the transactions input by the user and the output messages generated by the program, is shown.

```
01/22/75  LOGGED IN AT  23.48.10
          WITH USER- ID B1
          EQUIP/PORT 61/03
COMMAND- PSRCN

MJSCED PROGRAM READY.

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    2,010,1000,

010 - TRANSACTION APPLIED. BALANCE IS                          0000001005

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    3,020,3

   INVALID AMT. - TYPE 2 OR 3 TRANSACTION

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    3,020,3,

020 - TRANSACTION APPLIED. BALANCE IS                          0000000007

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    2,025,100,

025 - INVALID ACCOUNT NO.

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    1,025,

025 - TRANSACTION APPLIED. BALANCE IS                          0000000000

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    2,025,555,

025 - TRANSACTION APPLIED. BALANCE IS                          0000000555

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    4,030,

030 - DELETED FROM FILE.   BALANCE WAS                         0000000015

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    2,030,99,

030 - INVALID ACCOUNT NO.

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    4,060,
```

Figure 3-14. Printout of User's Terminal Input/Output (Sheet 1 of 2)

```
060 - DELETED FROM FILE.    BALANCE WAS                          0000000030

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    2,080,222,

080 - TRANSACTION APPLIED. BALANCE IS                            0000000262

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    1,085,

085 - TRANSACTION APPLIED. BALANCE IS                            0000000000

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    2,085,9999999999,

085 - TRANSACTION APPLIED. BALANCE IS                            9999999999

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    4,100,

100 - DELETED FROM FILE.    BALANCE WAS                          0000000050

ENTER NEXT UPDATE IN FORMAT  TYPE,ACCOUNT,AMOUNT,  OR END TO TERMINATE
                                    END

COMMAND- LOGOUT

CPA       .685 SEC.          .685 ADJ.
SYS TIME                     .848
CONNECT TIME      0 HRS.    8 MIN.
 01/22/75  LOGGED OUT AT 23.56.32.
```

Figure 3-14. Printout of User's Terminal Input/Output (Sheet 2 of 2)

Figure 3-15 shows the dayfile produced by execution of the sample muj program. The entries for the job count messages were generated by the COBOL muj program; all other entries in the dayfile were generated by INTERCOM and the operating system. The message REQUEST(SWAPIOF,*AX) was generated as a result of loading the muj program into the system; the messages JOB KILLED and JOB REPRIEVED were generated as a result of terminating the muj program.

The job counts reflect the total transactions processed by the muj program; they are routed to the dayfile each time the muj program is terminated. No transaction is counted twice. For example, if a transaction is counted as a deletion, it is not included in the applied count; likewise, a transaction that is included in the count of records added is not included in the applied count, although the message generated indicates that the record was applied.

```
23.48.30.  .P.PI.P.  PSRCN.
23.48.31.  .P.PI.P.  REQUEST(SWAPIOF,*AX)
23.48.31.  .P.PI.P.  ( SWAPIOF ASSIGNED TO EST 01 )
23.48.32.  B1B1IB1.  PSRCN.
23.52.13.   SYSTEM.  MT21 STAT = 2200
23.56.20.  .P.PI.P.  NUMBER OF TRANSACTIONS APPLIED  005
23.56.20.  .P.PI.P.  NUMBER OF RECORDS DELETED       003
23.56.20.  .P.PI.P.  NUMBER OF RECORDS ADDED         002
23.56.22.  B1B1IB1.  EXIT..PSRCN.
23.56.22.  .P.PI.P.  JOB KILLED
23.56.22.  .P.PI.P.  JOB REPRIEVED
23.56.22.  .P.PI.P.    (PREVIOUS ERROR CONDITION RESET)
23.56.23.  .P.PI.P.$JOB KILLED
23.56.23.  .P.PI.P.$MS        0  WORDS  (    6400 MAX USED)
23.56.23.  .P.PI.P.$CPA        .626 SEC.          .626 ADJ.
23.56.23.  .P.PI.P.$IO         .315 SEC.          .315 ADJ.
23.56.23.  .P.PI.P.$CM       23.932 KWS.         1.460 ADJ.
23.56.23.  .P.PI.P.$SS                           2.430
23.56.23.  .P.PI.P.$PP        7.664 SEC.     DATE 01/22/75
23.56.23.  .P.PI.P.$EJ END OF JOB, **
```

Figure 3-15. Dayfile Generated by Sample Program

DEBUGGING A COBOL MUJ PROGRAM

A COBOL muj program can be initially tested with SCED, or with DUMMUJ, a program included in the SCED package for debugging the program. DUMMUJ contains the same entry points as SCED, but SCED calls to DUMMUJ other than TERMIN, TERMOUT, and EXITMUJ are treated as dummy calls.

To test the program with DUMMUJ, the program must be loaded with DUMMUJ. The user must also enter the INTERCOM command CONNECT,INPUT,OUTPUT to equate these files to the user's terminal. (This command is not required after the program has been installed with SCED and the muj subroutines.) File access, terminal language, screen input/output formats, and most of the program logic can be debugged running under DUMMUJ.

Once the debug of the program has been completed using DUMMUJ, the program is installed without additional modification with SCED and the muj subroutines, and the program segments are stored on the system library using the EDITLIB utility program. The remaining program logic, which comprises the protection of data areas through interlocks and user assignments, can then be checked.

SCED causes a full field length dump, along with error messages and codes, to be routed to the output file if the program aborts after it is installed.

Testing a COBOL muj program with DUMMUJ is not necessary, although it is practical. If the program is initially debugged with SCED, the program must be reinstalled each time a correction is required.

PROGRAM TESTING WITH DUMMUJ

Two methods of combining DUMMUJ and the muj program are shown in figure 4-1.

Program A in figure 4-1 shows a job deck that can be used to combine the binaries of the COBOL muj program, of the COMPASS subroutines needed to attach and return the files, and of DUMMUJ. (It is assumed that DUMMUJ has been extracted from the INTERCOM program library and assembled, and the binary output has been placed on a user library, cataloged as DUMMUJ.) The combined binaries are finally cataloged as permanent file SCEDPF. Once the job deck in Program A is run, the user can test the program at the terminal by entering the following INTERCOM commands:

```
ATTACH,SCEDPF,ID=SCED
CONNECT,INPUT,OUTPUT
SCEDPF
```

The next response from the user depends upon information generated by the program.

Program B in figure 4-1 shows the INTERCOM commands necessary to compile, load, and test a muj program from a terminal. The example assumes the DUMMUJ binaries and the source programs for the COBOL compile and COMPASS assembly have been placed on the user library and cataloged.

A problem with the output buffer size can occur when the user is debugging a COBOL muj program with DUMMUJ. If the output buffer size indicator is set to a value less than the number of words in the output buffer during a terminal output operation, the output message is not displayed upon the user's screen because the message is dropped by INTERCOM for not being in the proper format; however, program processing continues. Terminal output normally prompts the action the user must take. If no output messages are displayed, the user has no way of knowing when a response is required. When this happens, the user should enter the signal to disconnect from the program, correct the program, and test again.

PROGRAM A – Sample Job Deck for Muj Program Assembly

```
DUMMUJ,MT1,T100.
ATTACH,DUMMUJ,ID=SCED.          ATTACH DUMMUJ BINARIES
REQUEST,SCEDPF,*PF.             REQUEST PERMANENT FILE FOR
                                   COMBINED BINARIES
COBOL,B=MUJBIN.                 COMPILE COBOL SOURCE PROGRAM
COMPASS,S=CPCTEXT,B=MUJBIN.     ASSEMBLE COMPASS SOURCE SUBROUTINES
COPY,DUMMUJ,MUJBIN.             STACK DUMMUJ BINARIES
LOAD,MUJBIN.                    LOAD COMBINED BINARIES
NOGO,SCEDPF.                    COMPLETE LOAD LINKAGES
CATALOG,SCEDPF,ID=SCED.         CATALOG COMBINED BINARIES
7/8/9

     COBOL SOURCE PROGRAM

7/8/9

     COMPASS SOURCE PROGRAM TO ATTACH AND RETURN PERMANENT FILES

7/8/9
6/7/8/9
```

PROGRAM B – Muj Program Assembly Using INTERCOM Commands

```
ATTACH,DUMMUJ,ID=SCEDPF.        ATTACH DUMMUJ BINARIES
ATTACH,PFSCD1,ID=SCEDPF.        ATTACH COBOL SOURCE PROGRAM
ATTACH,PFSCD2,ID=SCEDPF.        ATTACH COMPASS SOURCE SUBROUTINES
COBOL,I=PFSCD1,B=MUJBIN.        COMPILE COBOL PROGRAM
COMPASS,I=PFSCD2,B=MUJBIN.      ASSEMBLE COMPASS SUBROUTINES
COPY,DUMMUJ,MUJBIN.             STACK DUMMUJ BINARIES
CONNECT,INPUT,OUTPUT.           ASSIGN I/O FILES TO TERMINAL
MUJBIN.                         LOAD AND EXECUTE MUJ PROGRAM
```

Figure 4-1. Methods for Testing Muj Program with DUMMUJ

SCED INSTALLATION REQUIREMENTS

The following information must be supplied to SCED through SCED installation parameters when the COBOL muj program is installed with SCED and muj subroutines:

The size of the user area, in central memory words, and the number of buffers to be allocated in central memory for the user area. The larger the number of buffers allocated in central memory, the better the muj program can perform multi-user processing.

The maximum number of users that can be connected to the program at any time.

The number (n) of interlocks that the program requires. The maximum value that can be assigned to an interlock in the program is n-1.

The size of each output buffer and the number of each size to be allocated in central memory. The maximum value that can be assigned to a buffer identification number in the program is the number of output buffer types specified minus one.

SCED installation requirements are discussed in detail in the INTERCOM 4 Multi-User Job Capability reference manual.

ERROR CONDITIONS AND CODES

All fatal errors generated by a COBOL muj program installed with SCED are identified by a code number. Error codes in the range 00 through 49 are produced by muj subroutines as a result of system errors generated by muj subroutines or SCED; these codes are described in the INTERCOM 4 Multi-User Job Capability reference manual.

Error codes 50 and higher are generated by SCED as the result of errors encountered in the program. These SCED errors are coded by types, and are shown below.

Type	Codes
SCED call sequence errors	50-5x
Interlock assignment errors	60-6x
Terminal output errors	70-7x
Errors caused by a user overwriting sections of SCED or muj subroutines	80-8x

Errors represented by codes 80-8x are possible because SCED and the muj subroutines are contained within the field length of the muj program and are not protected from overwrite. Table 4-1 lists the error codes and the associated conditions that cause the errors.

RECOVERY TECHNIQUES FOR TERMINAL COMMUNICATION PROBLEMS

SCED performs the following procedures for recovery from typical situations that can arise during terminal processing. (BREAK refers to those cases where the user either is temporarily disconnected from INTERCOM or enters %ABORT.)

BREAK DURING TERMINAL INPUT – If a break occurs during terminal input, a situation that is commonly caused by a temporary disconnect of a user's terminal from INTERCOM. SCED reissues the input request that was pending at the time of the break. That is, SCED returns to the waiting-for-input state.

TABLE 4-1. ERROR CODES AND CONDITIONS

Error Codes	Error Condition
50	INIT was called out of sequence.
51	CONNECT was called out of sequence.
52	Terminal session section function was called out of sequence.
53	DISCON was called out of sequence.
54	EXITMUJ was called out of sequence.
60	Interlock number specified by GETINT call is outside range specified during installation.
61	Interlock specified by GETINT call is already in use.
62	Interlock number specified by RETINT call is outside range specified during installation.
63	Interlock number specified by RETINT call is an unreserved interlock.
64	Disconnect is not allowed while interlock is still reserved.
70	Buffer type identification number is outside range specified during installation.
71	Word count is greater than size specified for buffer type requested.
72	TIO error occurred. This error is usually caused by incorrectly formatted terminal output. For example, failure to terminate the buffer with a 12-bit zero byte in the low-order word position causes this error condition.
80	TIO error occurred on read.
81	No free user ordinal is available for new user.
82	Identification of new user is already in use.
83	Muj subroutines returned identification unknown to SCED.
84	Muj subroutines status code is not recognizable by SCED.

PERMANENT DISCONNECT FROM THE MUJ – If a user is permanently disconnected from the muj program abnormally, SCED releases the interlocks and user area assigned to the user and passes control to the paragraph name specified in the INIT call for disconnect processing. The SCED call to DISCON must be included in the disconnect processing.

BREAK DURING TERMINAL OUTPUT – If a break occurs during terminal output, SCED proceeds as if the output were successfully completed and returns to the program after the TERMOUT call that initiated the output.

STANDARD CDC CHARACTER SETS A

CONTROL DATA operating systems offer the following variations of a basic character set:

CDC 64-character set

CDC 63-character set

ASCII 64-character set

ASCII 63-character set

The set in use at a particular installation was specified when the operating system was installed.

Depending on another installation option, the system assumes an input deck has been punched either in 026 or in 029 mode (regardless of the character set in use). Under NOS/BE 1, the alternate mode can be specified by a 26 or 29 punched in columns 79 and 80 of the job statement or any 7/8/9 card. The specified mode remains in effect through the end of the job unless it is reset by specification of the alternate mode on a subsequent 7/8/9 card.

Graphic character representation appearing at a terminal or printer depends on the installation character set and the terminal type. Characters shown in the CDC Graphic column of the standard character set table are applicable to BCD terminals; ASCII graphic characters are applicable to ASCII-CRT and ASCII-TTY terminals.

NOTE

In the following chart, characters identified by the heading CDC GRAPHIC are applicable to BCD-CRT models 214-11, 214-12, 217-11, 217-12, 731-12, and 732-12.

Characters identified by the heading ASCII GRAPHIC are applicable to ASCII (CRT and TTY) as follows:

ASCII-CRT

217-13, 217-14, 731-12, 732-12

711-10

714

733-10

ASCII-TTY

Model 33, 35, or 38 Teletype

713-10

STANDARD CDC CHARACTER SETS

CDC Graphic	ASCII Graphic Subset	Display Code	Hollerith Punch (026)	External BCD Code	ASCII Punch (029)	ASCII Code	CDC Graphic	ASCII Graphic Subset	Display Code	Hollerith Punch (026)	External BCD Code	ASCII Punch (029)	ASCII Code
:†	:	00†	8-2	00	8-2	3A	6	6	41	6	06	6	36
A	A	01	12-1	61	12-1	41	7	7	42	7	07	7	37
B	B	02	12-2	62	12-2	42	8	8	43	8	10	8	38
C	C	03	12-3	63	12-3	43	9	9	44	9	11	9	39
D	D	04	12-4	64	12-4	44	+	+	45	12	60	12-8-6	2B
E	E	05	12-5	65	12-5	45	–	–	46	11	40	11	2D
F	F	06	12-6	66	12-6	46	*	*	47	11-8-4	54	11-8-4	2A
G	G	07	12-7	67	12-7	47	/	/	50	0-1	21	0-1	2F
H	H	10	12-8	70	12-8	48	(	(	51	0-8-4	34	12-8-5	28
I	I	11	12-9	71	12-9	49	)	)	52	12-8-4	74	11-8-5	29
J	J	12	11-1	41	11-1	4A	$	$	53	11-8-3	53	11-8-3	24
K	K	13	11-2	42	11-2	4B	=	=	54	8-3	13	8-6	3D
L	L	14	11-3	43	11-3	4C	blank	blank	55	no punch	20	no punch	20
M	M	15	11-4	44	11-4	4D	, (comma)	, (comma)	56	0-8-3	33	0-8-3	2C
N	N	16	11-5	45	11-5	4E	. (period)	. (period)	57	12-8-3	73	12-8-3	2E
O	O	17	11-6	46	11-6	4F	≡	#	60	0-8-6	36	8-3	23
P	P	20	11-7	47	11-7	50	[	[	61	8-7	17	12-8-2	5B
Q	Q	21	11-8	50	11-8	51	]	]	62	0-8-2	32	11-8-2	5D
R	R	22	11-9	51	11-9	52	%	%††	63	8-6	16	0-8-4	25
S	S	23	0-2	22	0-2	53	≠	" (quote)	64	8-4	14	8-7	22
T	T	24	0-3	23	0-3	54	→	_ (underline)	65	0-8-5	35	0-8-5	5F
U	U	25	0-4	24	0-4	55	∨	!	66	11-0 or 11-8-2†††	52	12-8-7 or 11-0†††	21
V	V	26	0-5	25	0-5	56							
W	W	27	0-6	26	0-6	57	∧	&	67	0-8-7	37	12	26
X	X	30	0-7	27	0-7	58	↑	' (apostrophe)	70	11-8-5	55	8-5	27
Y	Y	31	0-8	30	0-8	59	↓	?	71	11-8-6	56	0-8-7	3F
Z	Z	32	0-9	31	0-9	5A	<	<	72	12-0 or 12-8-2†††	72	12-8-4 or 12-0†††	3C
0	0	33	0	12	0	30							
1	1	34	1	01	1	31	>	>	73	11-8-7	57	0-8-6	3E
2	2	35	2	02	2	32	≤	@	74	8-5	15	8-4	40
3	3	36	3	03	3	33	≥	\	75	12-8-5	75	0-8-2	5C
4	4	37	4	04	4	34	¬	~(circumflex)	76	12-8-6	76	11-8-7	5E
5	5	40	5	05	5	35	; (semicolon)	; (semicolon)	77	12-8-7	77	11-8-6	3B

†Twelve or more zero bits at the end of a 60-bit word are interpreted as end-of-line mark rather than two colons. End-of-line mark is converted to external BCD 1632.

††In installations using a 63-graphic set, display code 00 has no associated graphic or card code; display code 63 is the colon (8-2 punch). The % graphic and related card codes do not exist and translations from ASCII/EBCDIC % yield a blank (55_8).

†††The alternate Hollerith (026) and ASCII (029) punches are accepted for input only.

GLOSSARY B

BUFFER TYPE IDENTIFICATION NUMBER – A unique number assigned by SCED to distinguish one buffer size from another. SCED assigns the numbers consecutively, beginning with zero, at installation time. SCED uses the number to allocate SCED output buffers used in terminal output operations. Users must ensure that a buffer type of a size large enough to hold the output is specified in the terminal output buffer area in the program before issuing the call to SCED for a terminal output operation.

COMMON DATA AREA – A data area that can be used by all users executing in a COBOL muj program at any time or place in the program. Common data areas are rare in a COBOL muj program.

COMMON DATA AREA PROTECTED BY INTERLOCK – A data area that does not belong to a single user, although any user can gain exclusive access to the area for a given period of time through the use of an interlock.

CONSTANT – A data area that is initialized to a value during program load and remains unchanged throughout program execution.

DUMMUJ – A program designed for the debug of a multi-user job as a single-user task. DUMMUJ contains the same entry points as SCED; however, all calls to entry points other than TERMIN, TERMOUT, or EXITMUJ are treated as dummy calls.

INITIALIZATION SECTION – The first section of a COBOL muj program. The initialization section contains functions that are common to all users. The section is executed when the first user connecting to a COBOL muj program causes the program to be loaded into central memory, and during program restart.

INTERLOCK – A SCED feature designed to protect data areas by allowing only one user at a time to execute in specified sections of a COBOL muj program. An interlock is either reserved or released during program execution. When an interlock is reserved, no user can proceed beyond a statement reserving the interlock until the interlock is released by the user for whom it is reserved.

LOCAL DATA AREA – A data area that is fully utilized between SCED points. No special protection is required of this data area because it is normally initialized to some value before it is used.

MUJ SUBROUTINES – A set of INTERCOM library subroutines that interface between INTERCOM and SCED. These subroutines handle such functions as task scheduling, CIO request for input/output, and monitoring waiting users.

MULTI-USER JOB – A program that can have a single copy in execution for any number of users at one time.

RANGE OF THE INTERLOCK – The sections of a COBOL muj program that lie between the statements reserving and releasing an interlock. This range can be a complex area of code because each interlock can be reserved and released a number of times in the program.

SCED – A program that provides a set of COBOL entry points that can be called by a COBOL muj program to interface with muj subroutines.

SCED POINT – A point in a COBOL muj program where an ENTER call is made to one of the SCED functions. When a SCED point is entered, SCED can schedule other users for processing while the SCED function is being completed for the user issuing the call.

TERMINAL INPUT – Data entered at a terminal and transmitted over communication channels to a COBOL muj program where it is processed as input data.

TERMINAL OUTPUT – Data generated by a COBOL muj program and transmitted over communication channels to a terminal, where it is displayed upon a screen or printed.

TERMINAL SESSION SECTION – The sharable section of a COBOL muj program.

USER AREA – A special data area, a copy of which is maintained by SCED for each user connected to a COBOL muj program. Each user has complete control over the information contained in this private user area.

WRAP-UP SECTION – The last section of a COBOL muj program. The wrap-up section is executed when the last user executing in the program issues the call to disconnect. This section is common in the same way that the initialization section is common to all users.

INDEX

COMMENT SHEET

TITLE: SCED User Guide for INTERCOM 4
Multi-User Job Capability

PUBLICATION NO. 60494800 REVISION A

This form is not intended to be used as an order blank. Control Data Corporation solicits your comments about this manual with a view to improving its usefulness in later editions.

Applications for which you use this manual.

Do you find it adequate for your purpose?

What improvements to this manual do you recommend to better serve your purpose?

Note specific errors discovered (please include page number reference).

General comments:

FROM NAME: _____ POSITION: _____

COMPANY NAME: _____

ADDRESS: _____

NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

FOLD ON DOTTED LINES AND STAPLE

STAPLE

STAPLE

FOLD

FOLD

FIRST CLASS
PERMIT NO. 8241

MINNEAPOLIS, MINN.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U.S.A.

POSTAGE WILL BE PAID BY

CONTROL DATA CORPORATION
Publications and Graphics Division
215 Moffett Park Drive
Sunnyvale, California 94086

FOLD

FOLD

STAPLE

STAPLE

CORPORATE HEADQUARTERS, P.O. BOX 0, MINNEAPOLIS, MINNESOTA 55440
SALES OFFICES AND SERVICE CENTERS IN MAJOR CITIES THROUGHOUT THE WORLD

LITHO IN U.

CONTROL DATA CORPORATION