60494700

CONTROL DATA CORPORATION

INTERCOM VERSION 4 MULTI-USER JOB CAPABILITY REFERENCE MANUAL

CDC® OPERATING SYSTEM:
NOS/BE 1

REVISION RECORD	
REVISION	**DESCRIPTION**
A	Original printing.
(11-1-75)	
B	Revised to correct technical errors. Reflects NOS/BE version 1.3 at PSR level 473.
(6-13-78)	
Publication No. 60494700	

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

Address comments concerning this manual to:

Control Data Corporation
Publications and Graphics Division
4201 North Lexington Avenue
St. Paul, Minnesota 55112

or use Comment Sheet in the back of this manual

Control Data Corporation
Printed in the United States of America

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
Front Cover	-	
Title Page	-	
ii	B	
iii/iv	B	
v/vi	B	
vii	B	
viii	A	
1-1	A	
1-2	A	
1-3	A	
1-4	A	
1-5	A	
2-1	A	
2-2	A	
2-3	A	
2-4	A	
2-5	B	
2-6	B	
2-7	A	
2-8	A	
2-9	A	
2-10	A	
2-11	A	
2-12	A	
2-13	B	
2-14	A	
3-1	A	
3-2	A	
3-3	A	
3-4	A	
3-5	A	
3-6	A	
4-1	A	
4-2	A	
4-3	A	

Page	Revision	Software Feature Change
5-1	A	
5-2	A	
A-1	A	
A-2	A	
B-1	A	
B-2	A	
B-3	A	
Index-1	A	
Index-2	B	
Index-3	A	
Comment Sheet	B	
Back Cover	-	

PREFACE

INTERCOM Version 4 contains a multi-user job capability that allows more than one terminal user to access a single copy of an executing program. A program written in COBOL Version 4 or COBOL Version 5, COMPASS Version 3, or FORTRAN Extended Version 4 uses subroutine calls to interface with tables and routines internal to INTERCOM that permit the multiple access.

In addition to calls directly to the multi-user job subroutines, Control Data Corporation provides a higher level call capability for COBOL Version 4 or COBOL Version 5 users known as SCED. SCED interfaces with the routines that in turn interface with INTERCOM.

SCED and the multi-user job capability of INTERCOM operate under the NOS/BE Version 1 operating system on the CONTROL DATA® CYBER 170, CYBER 70 Models 71, 72, 73, and 74, and 6000 Series Computer Systems.

This manual describes the calls to the multi-user job subroutines and SCED subroutines. The reader is assumed to be an experienced programmer, familiar with terminal communication as well as the host language, who is writing an application program to be installed as part of the system library. Related publications are listed below.

Control Data Publication	Publication Number
INTERCOM Version 4 Reference Manual	60494600
COMPASS Version 3 Reference Manual	60492600
COBOL Version 4 Reference Manual	60496800
COBOL Version 5 Reference Manual	60497100
FORTRAN Extended Version 4 Reference Manual	60497800
SCED User Guide for INTERCOM Version 4 Multi-User Job Capability	60494800
UPDATE Reference Manual	60449900
NOS/BE Version 1 Installation Handbook	60494300

This product is intended for use only as described in this document. Control Data cannot be responsible for the proper functioning of undescribed features or parameters.

CONTENTS

APPENDIXES

INDEX

FIGURES

TABLES

MULTI-USER JOBS 1

The multi-user job (muj) facility of INTERCOM 4 enables a program to provide processing for many terminal users at the same time. The interface between a muj program and INTERCOM is a set of subroutines called the muj subroutines. These subroutines schedule users for muj program execution by maintaining a scheduling queue, monitoring the current condition of each user in the queue, and returning the users to the program, one at a time, as they become ready for further processing. Although many users can call the muj program from their terminals and be active simultaneously, the muj program itself processes only one user at a time; the other users wait in the scheduling queue maintained by the muj subroutines.

In addition to scheduling and switching users, the muj subroutines perform input and output operations between the muj program and individual terminals, monitor completion of output operations, interface with INTERCOM tables, and provide error processing and recovery facilities.

Writing an application as a muj program might improve real-time performance (input/output operations can be occurring for one user while the muj program is executing central processor code for another), and might also reduce central memory requirements, since only one copy of the muj program is loaded no matter how many users call it.

Muj programs are more difficult to design and code than single-user jobs, and their interactive nature makes debugging more difficult; therefore, they are normally restricted to heavily used applications in which users remain active for a considerable period of time. An example of an existing muj program is the EDITOR utility of INTERCOM. Interactive data base processing programs are also good candidates for programming as muj programs.

A muj program differs from a single-user job in that the muj program must be installed in the system; it cannot be submitted for execution as part of a normal user job. A program that is not installed in the system cannot call the muj subroutines.

To install a muj program, an absolute binary of the muj program must be added to a system library, and entries must be made to INTERCOM tables so that INTERCOM can recognize the muj program's name. The system EDITLIB utility must be used to install the muj program, either as part of the deadstart file or directly into the running system. A system EDITLIB usually requires the consent or participation of the principal systems programmer responsible for the system. Procedures for installation are discussed in section 4.

SUBROUTINE CALLS

The muj subroutines can be called from programs written in COBOL, COMPASS, or FORTRAN. The entry point names for the subroutines might vary from one language to another because FORTRAN and COBOL can call preprocessors instead of the COMPASS entry point names. The basic functions of the muj subroutines, however, do not change. Parameters in the calls to the muj subroutines result in logically equivalent functions from all languages, even though the number of parameters in the call might vary.

When preprocessors are called by FORTRAN and COBOL, the preprocessors accept parameters in forms convenient to the high-level languages and reformat them before issuing calls to the COMPASS entry points. The preprocessor routines are discussed in section 2 with the muj subroutines for which they can be substituted.

In addition to the muj subroutines and preprocessors, a higher-level interface named SCED is available for the COBOL programmer. SCED enables the programmer to write a multi-user job as if it were a single-user job, and simplifies the complexities involved in programming with the muj subroutines. SCED is discussed in section 3.

The basic functions of muj subroutines are:

To maintain tables of the status of each user connected to the muj program.

To communicate with user terminals and the muj program.

To initiate input/output operations between the muj program and connected terminals at muj program request.

To schedule the users ready for further processing, and pass the user areas to the muj program upon request or in the order in which they occur in the scheduling queue.

To monitor users waiting for input/output completion and prepare the users for further processing.

To switch the user and associated user areas into or out of muj program execution at program request.

The four multi-user job subroutines that can be called by a program are listed below, along with their specific functions.

Subroutine	Function
INMUJ	Performs initialization for a muj program.
USER	Handles user scheduling and switching, as well as file action requests.
TIO	Performs terminal input/output operations.
MSYSERR	Generates system error messages and dumps.

One of the subroutines, MSYSERR, cannot be called directly from COBOL; however, the subroutine can be referenced in a COBOL program by writing a COMPASS subroutine that issues the calls to MSYSERR.

TYPICAL PROGRAM STRUCTURE

The general order of events in a muj program is as follows:

Initialization: A call to INMUJ initializes data areas (INMUJ is executed only once each time the muj is loaded).

Execution: Calls to USER cause users to be switched, and calls to TIO perform terminal input and output operations.

Termination: The muj does not terminate in the normal manner; rather, when INTERCOM detects that no users are attached to the muj program, execution is halted.

Recovery: If termination is due to an error, recovery processing is initiated, and an automatic call is made to MSYSERR to generate dumps.

INITIALIZATION

Certain areas, such as muj tables and user areas, are established in the field length of the muj program to facilitate communications among the muj program, the muj subroutines, and the terminals. Each muj program must have a muj table area, and might have a user area and other areas set up for individual user files. Allocation of these areas is the responsibility of the muj program; allocation can be accomplished through the DIMENSION statement in FORTRAN, the BSS statement in COMPASS, or Data Division entries in COBOL. Some of the areas, such as the muj table, are accessed only by the muj subroutines, even though the areas are allocated by the user.

Allocating muj Tables

The call to INMUJ must be the first call to muj subroutines executed. The call to INMUJ passes information to the muj subroutines about the muj table that is used internally by the muj subroutines TIO and USER. The muj table is subdivided into four parts, as follows:

The first part contains status and accounting information, by user, of all users attached to the program.

The second part contains pointers to each active user file and its file environment table (FET).

The third part contains pointers to user area buffers.

The fourth part contains an FET for each internal file used for input/output by the muj subroutines.

Assigning User Areas

The muj program can maintain information associated with an individual user by assigning a special area to each attached user. The area is called a user area buffer. Space for the user area buffer is defined in the muj program, and the size and number of buffers to be allocated within the program are passed to the muj subroutines in the INMUJ call.

If more users are attached to the program than the number of user area buffers allocated, muj subroutines maintain the excess user area buffers on disk storage. By swapping user area buffers between disk storage and the muj program, the muj subroutines ensure that the appropriate user area buffer is in the muj program whenever the corresponding user is scheduled for processing.

User Files

A muj program can allow individual user files to be accessed during program execution; however, a user's file is not attached automatically when a user calls the muj program. Files must be attached during program execution by a special call to the USER subroutine which equates the name specified by the user with a name already compiled into the program. Certain established procedures eliminate user conflict during individual file accessing:

Space must be allocated for a file environment table in the muj program for each user file the muj program can attach concurrently. If the muj program is to use the file attaching facility, it is suggested that space be allocated for two or three FETs.

Space must also be allocated in the muj table, by the INMUJ call, for an entry composed of one central memory word for each user file that can be concurrently attached to the muj program. The central memory word identifies the owner and FET address of each attached user file.

In response to calls with file action requests, the USER subroutine inserts the file name table (FNT) address of the file in the fifth word of the FET to distinguish the file name from possible duplicates. Input/output is directed to the FNT entry by the pointer in the FET when an input/output operation is performed on a user file.

If the muj program attaches its own files, the file names must not conflict with attached user files. One way of ensuring that no conflicts occur is to have the muj program attach all its own files in the initialization section of the program before any user files are attached. In this way, even user files of the same name cannot cause conflicts with muj program files.

Terminal Input/Output

Special six-word FETs must be allocated in the muj table for internal use by muj subroutines in performing terminal input/output operations. Only one input/output request can be in process for a terminal at one time. As soon as an input/output operation is completed for a terminal, the FET is released and the six-word area allocated to the FET can be assigned to another terminal input/output operation.

The sixth word of the special FETs used for terminal input/output is set with the user's identification, which is used to route messages to the proper terminal and identify the sending user to the muj program. The user identification is also used in creating and releasing the FETs, since all of the FETs created for terminal input/output have the same file name.

EXECUTION

A muj program is activated and/or associated with a user when the muj program name is entered as a command at a terminal. When this command is issued, the specified muj program is brought into execution if no other user is being serviced by the muj; if the muj program is already active, the terminal is placed in the scheduling queue and eventually given to the muj program for processing by a call to USER.

An activiated muj program can handle processing for any number of terminal users, up to the maximum defined by the muj program on the INMUJ call. The USER subroutine keeps track of each user attached to the muj program. The muj program calls the USER subroutine whenever a new user can be scheduled for program execution. USER schedules the next user or a specific user for program execution and ensures that the corresponding user area buffer is in central memory. A new user can also be requested when processing for the current user cannot proceed until some condition is satisfied. For example, it might be necessary to wait until input/output is completed, a file is attached, or a buffer is available. When a user

awaiting some condition is returned to the scheduling queue, the USER routine keeps track of the condition awaited, and does not return the user until the condition is fulfilled.

The muj program and individual users communicate through the TIO subroutine. When the muj program is ready to communicate with an individual user, a call is issued to TIO indicating whether the program is ready either to accept input data or to transmit output data. TIO takes care of the actual transfer of data to and from the user's terminal through INTERCOM buffers; thus, the muj program is immediately freed to continue processing for other users.

TERMINATION

A muj program cannot terminate in the normal manner. When no users are attached, INTERCOM terminates the muj program. No reprieve is possible from this termination.

Each user processing in a muj program can terminate at any time. Typically, the user detaches from the muj by entering the command that the muj program interprets as denoting end of processing for a terminal. The muj program then issues a call to USER to detach the user from the muj. For certain errors, the muj program itself might be terminated as a result of internal system checks; in this case, all users are detached from the muj program.

The muj program is terminated automatically when the last user processing the muj program is detached. INTERCOM initiates this termination, and the program ceases to exist as a task in the system. When another user enters its name as an INTERCOM command, the program is reinitialized as a new task in the system.

ERROR PROCESSING

MSYSERR muj subroutine can be called when an error is detected in the muj program. Parameters in the call indicate the action that is to be taken as a result of the error. Some of the actions the muj program can request are:

Send error messages and codes to all users.

Abort the entire muj program.

TRACE PRINTOUT

The muj subroutines can be compiled optionally to generate a trace printout as an aid in diagnosing muj program aborts. This trace printout is generated and included in the abort dump if a *YANK MDEBUG card is included in the UPDATE job that extracts the muj subroutines from the INTERCOM program library.

The trace printout is generated by FORTRAN PRINT statements; consequently the muj programmer must provide, at address x, a word containing the left-justified display code characters OUTPUT in bits 59-18, and the address of a file information table (FIT) for the file named OUTPUT in bits 17-0. Address x must satisfy the condition RA+2$\leqslant$x$\leqslant$RA+63 (decimal). None of the words preceding address x can be binary zero. Linkage is provided by including the file name OUTPUT on the PROGRAM statement of a FORTRAN program.

In order to handle an abort, INMUJ calls the operating system RECOVR routine. If the muj program that is being written also calls RECOVR, this call should be included after the muj program calls INMUJ. The last call to RECOVR is executed first, in effect stacking requests.

ACCOUNTING

Each time the USER subroutine returns a user to the muj program in a ready state, the subroutine increments two servicing counts. One count is maintained in an internal table associated with the user to record the number of times the muj provided processing for the user since the last accounting period. The second count is incremented to tally the total number of muj services for all users since the last accounting period.

When a user's servicing count reaches a threshold value, USER performs the following accounting procedure:

Reads the total central processor time used by the muj program since the last accounting period.

Computes the percentage of the total central processor time received by each user.

Distributes the charges to all users for an equal percentage of the total central processor time used by the muj program.

Reinitializes the user's servicing count and the total muj program counts to zero to restart the accounting cycle.

If the maximum authorized time limit allotted to a user is exceeded, the state parameter returned by the USER subroutine informs the muj program. The muj programmer is responsible for ensuring that the specific user is denied any further muj processing except as required for exiting from the muj program without drastic consequences; for example, a multi-user text editor program could be programmed to allow the user to save a file before forcing an exit.

MUJ SUBROUTINE CALLS 2

Calls can be made from a COMPASS or a FORTRAN program to all four muj subroutines: INMUJ, TIO, USER, and MSYSERR. A COBOL program can call only three of the muj subroutines: INMUJ, TIO, and USER. MSYSERR can be accessed by a COBOL program only by a call to a COMPASS subroutine, which in turn issues the call to MSYSERR.

COBOL and FORTRAN programs can also issue calls to preprocessors to communicate with the muj subroutines. The use of the preprocessors allows the COBOL or FORTRAN program to specify parameters in standard language formats. The preprocessors convert the parameters to the format required by muj subroutines before calling the muj subroutines, and restore the parameters to the standard language format before returning to the calling program.

FORTRAN can call the following preprocessor:

USERFO instead of USER.

COBOL can call the following preprocessors:

USERCO instead of USER.

INMUJCO instead of INMUJ.

TIOCO instead of TIO.

INMUJ SUBROUTINE

The INMUJ subroutine performs initialization for multi-user jobs. The call to INMUJ passes to the INMUJ muj subroutine information required to fill in the muj table used by the USER and TIO subroutines. The space for the muj table must be allocated by the muj program, but the table is used only by the muj subroutines. The muj table, along with other tables internal to the muj subroutines, is initialized when a call to INMUJ is issued.

Other initialization performed by INMUJ includes connecting the file used for terminal input/output, initializing the INTERCOM tables TERMIN and TERMOUT within the muj field length, and calling RECOVR to reprieve on all error conditions.

The INMUJ subroutine must be called by the muj program before any other muj subroutines are called; it can be called only once. The calling sequence and parameter formats for INMUJ and INMUJCO are shown in figure 2-1.

INMUJ PARAMETERS

Parameters in the INMUJ call specify information the muj subroutines require to initialize the program for terminal communication.

mujtbl Parameter

The mujtbl parameter specifies the beginning of a block within the program that is allocated for use by the muj subroutines. The length of the block depends on the value of the other parameters in the INMUJ call:

(nuserfile) + 6(ntermfile) + 4(nuser) + (narea) = length of mujtbl

Figure 2-2 shows how the muj subroutines use the area within the mujtbl block.

nuserfile Parameter

The nuserfile parameter specifies the maximum number of user files that can be attached simultaneously to the muj program. For each active file, mujtbl contains a one-word entry that identifies the file's FET address and the user associated with the file. The FET is defined elsewhere in the muj program.

ntermfile Parameter

The ntermfile parameter specifies the maximum number of terminal input/output FETs that can be used. The muj table must contain space for these FETs (6*ntermfile).

INMUJ Calling Sequence

FORTRAN:

```
CALL INMUJ (mujtbl, nuserfile, ntermfile, nuser, narea, larea, uarea)
```

COMPASS:

```
          SA1     INPAR
+         RJ      INMUJ
            :
INPAR     VFD     60/mujtbl,60/nuserfile,60/ntermfile
          VFD     60/nuser,60/narea,60/larea,60/uarea
          DATA    0
```

COBOL:

```
ENTER INMUJCO USING mujtbl, nuserfile, ntermfile, nuser, narea, larea, uarea.
```

INMUJ Parameter Formats

Parameters	FORTRAN	COMPASS	COBOL
mujtbl	array	BSS or other storage allocation instruction	table
nuserfile	integer variable or constant	integer	COMP-1
ntermfile	integer variable or constant	integer	COMP-1
nuser	integer variable or constant	integer	COMP-1
narea	integer variable or constant	integer	COMP-1
larea	integer variable or constant	integer	COMP-1
uarea	array	BSS or other storage allocation instruction	table

Figure 2-1. INMUJ Calling Sequence and Parameter Formats

location mujtbl			
	nuser	Status and accounting information supplied by user. (4 central memory words per possible user)	Length of nuser = nuser*4
	nuserfile	Pointers to FET and user for each attached user file. FETs for user files must be defined elsewhere in the muj program (1 central memory word per possible attached user file)	Length of nuserfile = nuserfile*1
	narea	Pointers to each user area buffer in central memory. (1 central memory word per possible user area buffer)	Length of narea = narea*1
	ntermfile	FET for each defined internal terminal input/output file. (6 central memory words per possible concurrent terminal input/output request)	Length of ntermfile = ntermfile*6

Figure 2-2. Structure of mujtbl

nuser Parameter

The nuser parameter specifies the number of users that can be in some state of execution within the muj program. For each active user, mujtbl contains a four-word entry to hold status and accounting information. When the number of users in the muj program is equal to nuser, other users are denied access; a message from INTERCOM instructs those users to try to call the muj program from their terminals at another time.

narea Parameter

The narea parameter specifies the number of user area buffers to be allocated in central memory. For each user area buffer that is swapped into the muj program mujtbl contains a one-word entry pointing to the buffer. If user area buffers are not used in the muj program, narea can be either zero or omitted.

larea Parameter

The larea parameter specifies the length of each user area buffer in central memory words. If user area buffers are not used in the muj program, larea can be either zero or omitted.

uarea Parameter

The uarea parameter specifies the name of the block of memory in the muj program provided for user area buffers. If user area buffers are not used in the muj program, uarea can be either zero or omitted.

INMUJ USE

The call to INMUJ is made to initialize the program and to pass to muj subroutines the location of the muj table, as well as the size of each table entry. The muj subroutines use the information passed by INMUJ for interaction with the muj program. Care should be taken in allocating space for the muj table because an inadequate amount of allocated space might slow program response time to an unacceptable level. The amount of space allocated in the muj table should increase as the number of terminal users increases.

USER SUBROUTINE

The USER subroutine provides scheduling and switching of users into the muj program, and performs file action requests such as attaching and detaching user files. The USER subroutine receives instructions from the muj program concerning actions to be taken and returns to the muj program the status of the actions requested.

The muj program issues a call to the USER subroutine when a user executing in the program cannot proceed until some event occurs that is external to the program execution. Execution of USER temporarily halts processing for the current user, and causes another user to be switched into the program for processing while the first user is waiting for an operation such as input from the terminal.

The information that the muj program can pass to the USER subroutine includes:

- The identification of the current user and the pointer to the user area associated with the current user.
- The action that must be performed before the user can continue processing in the muj program.
- The request for the scheduling of either a specific new user or any new user.

USER subroutine execution returns information to the muj program indicating the status of actions requested. The information that the USER subroutine can return to the muj program includes:

- The identification of the new user switched into execution and the location of the user area associated with the new user.
- The status of the new user; that is, whether the new user is attaching to the program; whether the new user is ready to continue processing; or, if a specific new user was requested, whether that new user is presently unavailable to continue processing.

Each time USER is called, the user currently processing in the program is switched out and another user is switched in. The user scheduled for execution by the muj subroutine is determined by the status of the users in relation to the actions requested. The program has the option of specifying a switch of a

particular user, however. When the program, rather than the muj subroutines, decides when a given user is to be switched in, the program is also responsible for checking whether a prior request has been completed. If a requested user is not ready to continue processing, the user and user area are not switched in. In this instance, only the status of the requested user is returned to the program.

USER execution switches in a new user and the associated user area. (The use of user areas is optional; whether or not they are employed depends on the needs of the program.) A pointer to the user area is returned to the program along with the identification of the user. The program should set this user identification as the old user parameter the next time USER is called to switch out that user.

Whether or not the old user's area is swapped out to mass storage when the user is switched out of execution depends on the program logic. If the user area contains an FET or a buffer for an input/output request currently in process, the user area must remain in central memory until that operation is completed; under these circumstances the call to USER should indicate that the user area is to be retained in memory.

The calling sequences and parameter formats for USER, USERFO, and USERCO are shown in figure 2-3.

USER PARAMETERS

The parameters specified as part of the USER call identify the old user, specify the action to be performed, point to a buffer, and, in the case of a file action request, specify the user file name. On return from USER, parameters are set to show the results of execution of this call.

olduser Parameter

The olduser parameter contains the identification of the user that is to be switched out of execution. Two left-justified zero-filled display code characters identify a user. The identification is determined by INTERCOM and returned to the program in the newuser field each time a user is switched into execution. When the program calls USER for the first time, the olduser parameter should be set to 0.

actn Parameter

The actn parameter specifies the action to be performed. Most actions define the conditions under which the old user currently executing can be returned to the muj program for further processing.

The sign of the actn value specifies whether the user area can be swapped out to mass storage when the old user is switched out: a negative value retains the user area within the muj program field length; a positive value allows the user area to be swapped out. If user areas are not employed by the muj program, the sign of actn is irrelevant.

The actions are represented by the following codes:

0 Switch in a new user. Use 0 only when the muj program has no old user to be switched out.

1 Return the old user only when terminal input is available.

2 Return the old user at any time.

3 Return the old user only if the complete bit in the complt parameter word is set, indicating that a previously requested action is completed.

5 Detach the old user from the muj program. No further processing for this user is possible, since INTERCOM releases all internal tables associated with the user and reassociates all files with the terminal instead of the muj program.

6 (Reserved for INTERCOM EDITOR.)

7 Return the old user only when an output request initiated by a call to TIO is completed.

9 Return the old user only after all output reaches the terminal. Normally this actn is specified only as a follow-up to receipt of a state value with bit 57 set, which indicates a backlog of output for the user.

11 Create a new local file for the old user; the fetadd (or, in a compiler language, fname) parameter specifies the file name. Return the old user when the new file is created.

USER Calling Sequence

FORTRAN:

```
CALL USERFO (olduser, actn, state, newuser, iarea, lwait, flun, fname)
```

COMPASS:

```
          SA1     USPAR
          RJ      USER
          :
USPAR     VFD     60/olduser,60/actn,60/state
          VFD     60/newuser,60/iarea,60/complt,60/fetadd
          DATA    0
```

COBOL:

```
ENTER USERCO USING olduser, actn, state, ibreak, iperms, newuser, iarea, fwait, fname,
itime, ibklg.
```

USER Parameter Formats

Parameters	FORTRAN	COMPASS	COBOL
olduser	integer variable; left-justified, zero-filled	display code characters; left-justified zero filled	COMP, SYNC LEFT
actn	integer variable	integer	COMP-1
state	integer variable	integer	COMP-1
newuser	integer variable; left-justified, zero-filled	display code characters; left-justified, zero filled	COMP-1 SYNC LEFT
iarea	integer variable or constant	integer	COMP-1
complt	— —	address	— —
fetadd	— —	address	— —
lwait	integer	— —	— —
flun	integer variable	— —	— —
fname	file named on PROGRAM statement	— —	file named in FD entry
ibreak	— —	— —	COMP-1
iperms	— —	— —	COMP-1
fwait	— —	— —	display code
itime	— —	— —	COMP-1
ibklg	— —	— —	COMP-1

Figure 2-3. USER Calling Sequence and Parameter Formats

12 Attach an existing local file for the old user; the fetadd (or fname) parameter specifies the file name. Return the old user when the local file associated with the terminal is attached to the muj program.

13 Delete a local file for the old user; the fetadd (or fname) parameter specifies the file name. The actn parameter results in the performance of an operating system UNLOAD function. Return the old user after the file is deleted.

14 Detach an existing file and make it a local file; the fetadd (or fname) parameter specifies the file name. Return the old user after the attached file is made a local file.

16 Attach the local file associated with the old user if it exists; otherwise, create a local file for the old user. The fetadd (or fname) parameter specifies the file name. Return the old user after the local file is attached.

state Parameter

The state parameter is returned to the muj program by USER execution. It represents the status of the new user being returned to the muj program for processing or it specifies the reason why a requested user cannot be returned. Table 2-1 lists the values returned in the state parameter.

In calls to USER and USERFO, the state value is returned in bits 0-17. If the status is concerned with a permanent file, bits 18-22 are set to one to indicate the permissions granted for the file:

bit 18 Read permission is granted.
bit 19 Extend permission is granted.
bit 20 Modify permission is granted.
bit 21 Control permission is granted.
bit 22 Permanent file is attached.

In calls to USER and USERFO, flags are returned in bits 57-59:

bit 57 The user indicated by the newuser parameter has an output backlog within the INTERCOM buffer; unless USER is called with actn=9, subsequent output to this user will be delayed.

bit 58 The user indicated by the newuser parameter exceeded the maximum time limit as specified in the INTERCOM password file; the muj program should call USER with actn=5 to detach the user from the muj program.

bit 59 The user newuser entered a break (%A, ESC A or CTRL-Z characters) or a temporary terminal disconnect occurred. See USER Subroutine Use in this section.

In calls to USERCO, the only values returned in state are those listed in table 2-1. Four additional parameters, iperms, ibklg, itime, and ibreak, contain the permanent file and flag information.

iperms Parameter

The iperms parameter contains permanent file information corresponding to bits 18-22 of the state parameter in a USER or USERFO call. The iperms integer parameter is valid only in a call to USERCO.

ibklg Parameter (Optional)

The ibklg parameter contains the output-backlog information corresponding to bit 57 of the state parameter in a USER or USERFO call. The ibklg parameter is valid only in a call to USERCO.

itime Parameter (Optional)

The itime parameter contains the session time limit exceeded information corresponding to bit 58 of the state parameter in a USER or USERFO call. The itime parameter is valid only in a call to USERCO.

ibreak Parameter

The ibreak parameter contains the break information corresponding to bit 59 of the state parameter in a USER or USERFO call. The ibreak parameter is valid only in a call to USERCO.

newuser Parameter

The newuser parameter specifies the new user's identification. Its meaning depends on whether USER is being called or has already been called. When USER is being called, the newuser parameter identifies the new user to be switched into the program:

0 Switch in any user.
≠0 Switch in only the user identified.

When USER execution is complete, the newuser parameter is returned to the muj program. If a user was switched in, the newuser parameter identifies that user. If state=4 or 20-30, a new user was not switched in, but the newuser parameter indicates the user for which status information is being returned in the state field.

TABLE 2-1. state VALUES RETURNED TO MUJ PROGRAM FROM USER CALLS

state	Old User Out	Last actn for New User	New User In	Comment
0				state occurs only when the muj program is scheduling user for processing. If the program requests a user currently in execution, an error condition exists. If the program requests a user but that user has never been switched out, however, state=0. This never occurs as long as the program sets the olduser field to switch out the current user before a new user is switched in.
1	yes	1 (input available)	yes	An actn=1 request is fulfilled.
2	yes	2 (any ready user)	yes	
3	yes	3 (complt parameter set)	yes	An actn=3 request is fulfilled.
4	no	any	no	The value passed to USER in the olduser field does not correspond to the identification of any user connected to the program.
5	yes	(not applicable)	yes	The user is executing in the muj program for the first time.
6	- -	- -	- -	(Reserved for INTERCOM EDITOR.)
7	yes	7 (output buffer free)	yes	TIO output request complete.
8	yes	any	yes	INTERCOM logged out the user after the user failed to log back in within an installation-specified time period and following a line disconnect. The user is switched in so that the program can clear tables or perform other wrap-up functions and disconnect the user.
9	yes	9 (wait for backlog to clear)	yes	An actn=9 request is fulfilled.
10	yes	11 or 16 (attach local file)	yes	An actn=11 or actn=16 request is fulfilled.
11	yes	11 (attach local file)	yes	An actn=11 request was not fulfilled because the user already has a file by that name.
12	yes	12 (attach file to muj program)	yes	The file was not attached because no local file with the name specified exists for that user terminal.
13	yes	12 or 16 (attach file to muj program)	yes	The file was not attached because the user previously referenced the file in an INTERCOM CONNECT command that caused the file to be connected to the terminal. The file cannot be attached to the muj program until the user enters an INTERCOM DISCONT command to disconnect the file from the terminal.
14	yes	13 (delete file)	yes	An actn=13 request is fulfilled.
15	yes	11 or 16 (create new local file)	yes	An actn=11 or actn=16 request is fulfilled.

TABLE 2-1. state VALUES RETURNED TO MUJ PROGRAM FROM USER CALLS (Continued)

state	Old User Out	Last actn for New User	New User In	Comment
16	yes	14 (detach and make local file)	yes	An actn=14 request is fulfilled.
17	yes	12 (attach local file)	yes	The file was not attached to the muj program because the limit established by the INMUJ call was reached.
18	yes	11 or 16 (make local file)	yes	The file was not made local because the INTERCOM limit for the maximum number of local files was reached.
20	no	none	no	The user identified in the newuser field is not attached to the muj.
21	no	1 (input available)	no	The user identified in the newuser field is not ready for execution because no terminal input is available.
22	no	none	no	No user area buffer is free to be assigned to newuser. The muj program must process at least one other user and return a user area from memory before this user can be requested again.
23	no	3 (complt parameter set)	no	The user identified in the newuser field is not ready for execution because the complt (fwait or lwait in FORTRAN and COBOL) parameter has not been set.
24	no	9 (wait for backlog to clear)	no	The user identified by the newuser field is not ready for execution beacuse some output data still remains in the INTERCOM buffer.
25	no	7 (wait for buffer)	no	The user identified by the newuser field is not ready for execution because an output buffer is not available.
26	no	any	no	The user identified by the newuser field is not ready for execution because the associated user area is now being rolled to mass storage.
27	no	any file action code	no	The user identified by the newuser field is not ready for execution because the file action request is not complete.
28	no	13 or 14 (delete or detach file)	no	The user identified by the newuser field is not ready for execution because the file detach or delete request is not complete.
29	- -	- -	- -	(Reserved for INTERCOM EDITOR)
30	no	5 (detach user)	no	An actn=5 request is in process.

iarea Parameter

The iarea parameter provides a user area pointer. Its meaning depends on whether USER is being called or has already been called. If the muj program has not defined user areas in the INMUJ call, the iarea parameter is ignored.

When USER is being called, the iarea parameter identifies the location of the user area for the old user to be switched out. Most often the program simply respecifies the location received by the program, and does not need to use the pointer during execution.

When USER execution is complete, the iarea parameter identifies the location of the user area for the new user switched in. If no user was switched in, iarea remains as it was when USER was called.

complt Parameter

The complt parameter is valid in a call to USER only. It specifies the address of a central memory word, residing anywhere in the muj field length, in which bit 0 will be set by some external program (such as the operating system input/output routines) to indicate the completion of an operation. The complt parameter is meaningful only when actn=3, but it must be specified for all file action requests (act=11-16).

fetadd Parameter

The fetadd parameter is valid in a call to USER, but not USERCO or USERFO. It specifies the file environment table location. This parameter is meaningful and need be specified only for USER calls with actn=11-16.

If USER is being called with actn=1, actn=12, or actn=16 (create or attach file to muj program), the first word of the FET must specify the logical file name in display code, left-justified and zero-filled. When the user is again switched in for processing, the fifth word of the FET contains the file name table pointer for the file in bits 48-59.

If USER is being called with actn=13 or actn=14 (delete or detach file), the fifth word of the FET must specify the file name table pointer in bits 48-59.

flun Parameter

The flun parameter is valid in a call to USERFO only. It specifies the logical unit number of a file to be processed that is identified on the PROGRAM statement of the muj program. The parameter is meaningful only when actn=11-16.

fname Parameter

The fname parameter is valid in a call to USERFO or USERCO only, and is meaningful only when actn= 11-16. The fname parameter is the name of a file in display code, left-justified and zero-filled, which identifies the file to be processed.

In a call to USERCO, fname must correspond to a file name referenced in an ASSIGN clause in the Data Division; otherwise an error condition exists.

In a call to USERFO fname can be any file name, whether or not it is referenced in a PROGRAM statement. If the fname parameter is the name of a file on the PROGRAM statement, that file is used to carry out the file action request. If the fname parameter is not the name of a file on the PROGRAM statement, the file name is equivalenced to the file identified by the flun parameter.

fwait Parameter

The fwait parameter is valid only in calls to USERCO. It is meaningful only when actn=3, but must be specified for all actn=11-16 requests, also. For an actn= 11-16 request, the fwait parameter is the same as the fname parameter. The fwait parameter is the logical equivalent of the complt parameter used in a COMPASS call to USER. In a call to USERCO it specifies the name of a file in display code, left-justified and zero-filled; the file name must have previously been referenced in an ASSIGN clause in the Data Division.

lwait Parameter

The lwait parameter is valid only in calls to USERFO. It is meaningful only when actn=3, but must be specified for all actn=11-16 requests as well. The lwait parameter is the logical equivalent of the complt

parameter used in a COMPASS call to USER. In a call to USERFO it specifies the name of a file in display code, left-justified and zero-filled; the file name must have previously been referenced in a PROGRAM statement.

USER SUBROUTINE USE

The muj program should always check the state of a new user returned to it for processing, to determine if a break has been entered or a temporary disconnect has occurred. The muj program must determine the meaning of the break and the processing required for it. Entry of the break at a terminal terminates output currently being sent to the terminal and unsets certain wait conditions in effect for the user; for instance, if the user is waiting for input when the break is entered, the muj program must again request that the user be switched out to wait for input, even though bits 0-17 of state indicate input is ready.

When the USER subroutine returns newuser for processing, it performs only a normal subroutine return. In many cases, the muj program branches to a particular part of the program. The part depends on the user status and actn values. If user areas are employed, branching can be accomplished by maintaining a reentry address in the user area. If a reentry address is kept, care must be taken that associated muj routines are reentrant, since other information, including return addresses for nested subroutines, might have to be saved in the user area, and the passing of parameters in reentrant subroutines might have to be avoided.

If they are employed, the user area buffers can be the biggest obstacles in user scheduling. Even if a user is ready for processing, a user area buffer must be available before the user can be passed to the muj program for processing. An adequate number of user area buffers should consequently be allocated, and information for a specific user should not be retained in central memory in the user area buffers during a user switch unless retention is absolutely necessary. The use of FETs or input/output buffers in the user areas should be avoided, if possible.

TIO SUBROUTINE

The TIO subroutine performs terminal input/output for a specific user. If a call to TIO specifies a user that is currently processing an input/output operation, the request is rejected and an error code is returned in a TIO parameter.

All terminal input/output in a muj program must be requested through the TIO subroutine. The calling sequence and parameter formats for TIO are shown in figure 2-4.

TIO PARAMETERS

A call to the TIO subroutine specifies four parameters identifying the user and the data to be transmitted. A fifth parameter is returned to the program if an error occurs.

user Parameter

The user parameter specifies the identification of the user for whom an input/output operation is requested. The value of the user parameter is normally obtained from information returned to the muj program by the USER subroutine.

iocode Parameter

The iocode parameter specifies the type of input/output operation requested. The values used for iocode and an explanation of what each value indicates are shown below.

1 Perform a read operation from a terminal. The request should be made only after the USER subroutine indicates that input from the terminal is ready for a user. Control is not returned to the muj program until the data requested by the read operation is transferred into the program at the location indicated by the tiobuff parameter.

2 Perform a write operation to a terminal. Control is returned to the muj program immediately after the write operation is initiated and the area indicated by the tiobuff parameter is free. The FET or buffer might not be free at the time control is returned to the muj program.

TIO Calling Sequence

FORTRAN:

```
CALL TIO (user, iocode, tiobuff, leng, err)
```

COMPASS:

```
          SA1     TIPAR
          RJ      TIO
           :
           :
TIPAR     VFD     60/user,60/iocode,60/tiobuff,60/leng,60/err
          DATA    0
```

COBOL:

```
ENTER TIOCO USING user, iocode, tiobuff, leng, err.
```

TIO Parameter Formats

Parameters	FORTRAN	COMPASS	COBOL
user	integer variable with two left-justified display code characters	left-justified; two display code characters	COMP, SYNC LEFT
iocode	integer variable	integer	COMP-1
tiobuff	array	BSS or other storage allocation instruction	table
length	integer variable	integer	COMP-1
err	integer variable	integer	COMP-1

Figure 2-4. TIO Calling Sequence and Parameter Formats

3 Perform a one-line read operation from a terminal. The code differs from iocode=1 in the one-line limitation. A line of data is transferred to the muj program at location tiobuff; any other input for the user remains in the INTERCOM buffers and can be read later.

4 Perform a READSKP operating system function from a terminal. If the first line of terminal input does not fit into the circular buffer, data is read until the buffer is full; the remainder is discarded. No error code is returned in the FET. If the first line of data fits into the circular buffer, the code acts as a normal read (iocode=1).

5 Perform a one-line READSKP from a terminal. If the line does not fit into the circular buffer, data is read until the buffer is full; the remainder is discarded. No error code is returned in the FET. If the line fits into the circular buffer, the code acts as a normal one-line READSKP (iocode=3).

tiobuff Parameter

The tiobuff parameter specifies an array or table that is to be used in a terminal input/output operation. When a read operation is specified, tiobuff receives the input data; when a write operation is specified, the data is written from tiobuff. For a write

operation, tiobuff must contain the data that is to be transmitted to the terminal in display code format, the carriage control characters, and the end-of-line indicator. The end-of-line indicator must contain 12 bits of binary zeros in bits 0-11 of a central memory word. The carriage control characters are listed in the INTERCOM reference manual.

leng Parameter

The leng parameter specifies the number of words of information to be either received in tiobuff or transmitted from tiobuff. After a read, the leng parameter is set to the number of words transmitted to tiobuff. If the number of words specified in a read operation cannot be accommodated in tiobuff, the input data is truncated. If the number of characters of data involved in the input/output operation is either one less than an even multiple of ten or an exact multiple of ten, leng must include an extra word to contain the end-of-line indicator; otherwise the end-of-line indicator is placed in bits 0-11 of the last word of output data.

err Parameter

The err parameter is a field that is set for the muj program's use after a return from TIO execution. It indicates errors that occurred during input/output processing. The parameter values are shown below.

0 No error occurred.

1 An input/output operation is in process for the specified user; the requested operation cannot be performed. This condition should not arise if user switching is done properly.

2 An error was made in the TIO call. The error is usually an illegal value for iocode.

3 No terminal FET was available for the input/output request; it must be reissued later.

4 A read operation was requested, but input was not ready for the user.

5 The user identification is not that of a user attached to the muj program.

6 The buffer is shorter than the input; the input is truncated accordingly.

7 Information in the buffer is improperly formatted; an end-of-line zero byte is not in the last word.

8 A TIO system error occurred.

TIO USE

The muj program must format the data to be sent to the terminal in display code, with a carriage control character and an end-of-line indicator (a 12-bit byte of binary zeros in bits 0-11 of a central memory word). FORTRAN programs can use ENCODE to format output messages; COBOL can use MOVE or some other type of editing operation. The programmer must ensure the presence of the binary zero end-of-line indicator in the last word to be sent to the terminal, however.

Since the FETs used for terminal input/output are in the area defined by mujtbl in an INMUJ call, a user area need not be retained in central memory during terminal input/output unless the input buffer is in the user area.

If an input/output request is rejected because no terminal FET is available, a user switch should be requested, with the current user to be returned under no special conditions (actn=2). The input/output request can then be reissued.

A read request, issued by calling subroutine TIO with iocode set to the type of read desired, causes terminal input data to be transferred from an INTERCOM buffer, where it is received from a user terminal, to the muj program buffer indicated by the tiobuff parameter. A read request should be issued only after the USER subroutine informs the muj program that input is ready for a specific user; that is, after USER returns a new user to the muj program with state=1. The muj program can ensure that USER returns a specific user to the muj program only when the user's input is ready by returning the user as olduser with actn=1, in a call to USER.

A write request, issued by calling subroutine TIO with iocode=2, initiates data transfer from a muj program buffer to an INTERCOM output buffer, where it is

sent to the user's terminal. The muj program considers output complete when all data is removed from the muj program buffer and the buffer is free for use in another operation.

If a muj program generates output for a specific user faster than it can be displayed at the user's terminal, the output accumulates in INTERCOM's output buffers as backlog. When the backlog becomes excessive, INTERCOM eventually refuses to accept additional output from that user until the backlog is diminished.

The muj program must return users to USER with actn=7 after a write request to ensure that output is completed before attempting further processing for that terminal. This in turn might tie up the output buffer indefinitely; the following actions can be taken after the user is returned from the USER call, to determine whether a danger of a backlog exists on the next write request:

Examine bit 57 when the user is returned to the muj with state=7; state=7 indicates that the muj buffer used for output is free. If bit 57 is set, the user's output is backlogged and the next write statement for the user will probably be refused.

When bit 57 is set, the muj program can return the user to USER with actn=9 to wait until the backlogged output reaches the user's terminal. Returning the user with actn=9 ensures acceptance of the next write request and earlier availability of the buffer holding the output data.

MSYSERR SUBROUTINE

The MSYSERR subroutine generates system error messages and requests dumps for the muj program. MSYSERR is also called internally by the muj subroutines to generate error messages and dumps when errors are detected through internal system checks; however, the muj program has no control over the errors generated by the muj subroutines. Any dump requested is written to the file OUTPUT in standard dump format. OUTPUT is disposed immediately to the central site printer. The calling sequence and parameter formats for MSYSERR are shown in figure 2-5. A calling sequence is not supplied for COBOL. A COBOL muj program can call MSYSERR from a COMPASS subprogram by using the COMPASS calling sequence.

The error messages and codes generated by the muj subroutines are shown in Appendix B.

MSYSERR PARAMETERS

The first two parameters, number and iflags, are required to specify the error number and type of diagnostic information to be generated. The auxsub parameter is optional; it identifies an error routine.

number Parameter

The number parameter provides the system error number that appears in messages and dumps, as shown below.

0–49	Reserved for the muj subroutines system errors.
50–59	Available for the muj program errors.

iflags Parameter

The bit-structured iflags parameter specifies the diagnostic information to be generated by MSYSERR. The bits and the meaning associated with each setting are:

0-3	Dump to be made (octal value)	
	0	No dump is to be made.
	1	Dump the muj field length.
	2	Dump all of the field length and low core including the INTERCOM area.
	3	Dump the muj field length, low core (including the INTERCOM area), and the RBT area of high core.
4-5	Reserved (should be 0)	
6	Action after execution of MSYSERR	
	0	Return to the muj program.
	1	Abort the muj program.
7	User messages	
	0	Send no messages.
	1	Send a MUJ SYSTEM ERROR message to all users.

MSYSERR Calling Sequence

FORTRAN:

```
CALL MSYSERR (number, iflags, auxsub)
```

COMPASS:

```
          SA1     MSPAR
          RJ      MSYSERR
           .
           .
           .
MSPAR     VFD     60/number,60/iflags,60/auxsub
          DATA    0
```

MSYSERR Parameter Formats

Parameters	FORTRAN	COMPASS
number	integer variable or constant	integer
iflags	octal value	bit string
auxsub	external subroutine name	label

Figure 2-5. MSYSERR Calling Sequence and Parameter Formats

8 Dayfile messages

- 0 Send no messages.
- 1 Send a system error message to the dayfile.

9 Reserved (should be 0)

10-59 Unused

auxsub Parameter

The auxsub parameter names a subroutine that MSYSERR calls immediately before disposing the file OUTPUT to the central site printer.

MSYSERR USE

The reentrant nature of a muj program renders debugging difficult. Because a muj program services many users simultaneously, an abort affects all users; therefore, it is especially important to make sure a muj program is essentially error-free before it is installed in the system library.

The muj subroutines contain a number of internal checks for system errors that determine whether to abort the user command or the entire job. The muj program can also use a similar decision-making process to generate error messages and codes. When a system error is encountered in the muj program, the program must choose one of the following courses of action:

Ignore the situation and continue processing.

Issue messages and/or request dumps indicating that the error occurred, and either abort the command of the user currently being processed, or abort the entire muj.

SCED INTERFACE TO MUJ SUBROUTINES 3

SCED is a set of interface routines that provides a means to write multi-user jobs in COBOL that is simpler to use than the calls to the muj subroutines. Applications such as data base access are particularly suited to use of SCED; however, programmers writing more generalized applications may find that SCED does not offer the flexibility that direct calls to the muj subroutines provide.

The philosophy of a program using SCED differs in several respects from that of a program calling the muj subroutines directly. A COBOL muj program that interfaces with SCED is written as if it were to service a single terminal. Program logic assumes that the flow of control through the program is similar for all users: connect the user to the program, input data from the terminal, process the data, send output to the terminal, and, when all dialogue with the terminal is complete, disconnect the user from the program. Each time the program issues a call to SCED to perform terminal input/output or another SCED function, the program relinquishes control to the SCED routines. SCED schedules all users into execution of the program; and, of more significance from a programming standpoint, returns each user to the program immediately following the SCED call at which that user left execution. The SCED routines can switch a user out of execution and put another user into execution at the same or a different area of the program.

SCED functions that are performed after the muj program relinquishes control include:

Tracking the stage of processing of the individual users so that each user executes the complete program.

Requesting execution of the muj subroutines and processing responses from those subroutines.

Maintaining and allocating buffers for terminal input/output.

Processing permanent and temporary terminal disconnects and user aborts.

Another feature that SCED offers the application programmer is a debugging aid called DUMMUJ. DUMMUJ contains the same entry points as SCED, but calls to the SCED functions other than for terminal input/output are treated as dummy calls. Much of the muj program can be tested by executing the DUMMUJ routines rather than SCED routines. Once the single-user logic is debugged, the program should be installed with SCED replacing DUMMUJ for final checkout of conditions such as protection of data areas and interlock handling.

A program calling SCED, like a program calling the muj subroutines directly, must be installed as part of the system before it can be used from a terminal. Section 4 presents a typical installation procedure.

SCED USE IN A PROGRAM

The entry points to SCED, along with the action each achieves, are as follows:

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.

Each time a SCED call is made, the program relinquishes control to SCED. SCED returns control to the program for a user at the statement after the SCED call. Control is returned in such a manner that the continuity of processing for the user switched into execution is maintained.

PROGRAM ORGANIZATION

The Procedure Division of the application program is usually organized in three parts: initialization, terminal session, and termination.

The initialization portion of the program is executed when the program is first brought into central memory; it includes such activities as attaching the files that the terminal users access. The first call to SCED in the program, INIT, causes the SCED subroutines to initialize areas in the program field length that are to be processed by SCED during subsequent program execution. INIT is executed only once, by the first user to enter a muj program.

Terminal session processing begins with a call to the SCED function CONNECT. Each new user entering the program begins execution at this point. The statements that fall between the call to CONNECT and the call to DISCON make up the terminal session portion of the program. Terminal session processing is the only part of the muj program that is executed by all users.

Calls to SCED functions TERMIN, TERMOUT, GETINT, RETINT, and IOWAIT are made during terminal session processing. SCED maintains the continuity of execution for a single user, although many other users can execute in the muj program between executions for the first user.

The termination portion of a program, like the initialization code, is executed only once for each loading of the muj program. Any files attached during initialization must be closed and detached in the termination of the program. The last statement in the muj program must be a call to EXITMUJ, which releases the muj program from the system.

If a terminal user is permanently disconnected from the program by a line disconnect, SCED returns control to the muj program at the paragraph name specified in the INIT call. The paragraph must contain a call to DISCON to disconnect the user from the program. SCED itself releases any assigned interlocks and frees the user area.

A break, which is caused by a temporary line disconnect of a terminal or a %ABORT entered by a terminal user, need not be handled within the muj program because SCED performs recovery processing. A break during terminal input causes SCED to discard what was input before the break and return to the waiting-for-input state; a break during terminal output causes SCED to return control to the statement after the TERMOUT call as if all the output were successfully completed.

PROGRAMMING CONSIDERATIONS

Some special considerations must be made in a COBOL program that interfaces with SCED.

Input/Output File Processing

All files accessed from a COBOL program must be identified in the Environment Division. Although the muj subroutines provide a mechanism for equating a user-specified file name with a file name defined in the Environment Division, the SCED routines do not; consequently, all files accessed by any user connected to the muj program must be defined by the COBOL program. Further, the program itself must provide for attaching those files.

Only mass storage files can be accessed in a program calling SCED. Most often they are permanent files in existence before the program is executed. The permanent files must be attached to the program by a subroutine that can issue an RA+1 call, and opened by an OPEN statement, before any user is connected to the program. In the termination portion of the program, the files are closed and then returned to the operating system by another subroutine that issues an RA+1 call.

User Areas

A user area is a special data area defined within the program. Any information within a user area is associated with one and only one user. Each time a user executes a SCED point and is taken out of

execution, the information in the user area is saved. When that user returns to execution, the information is copied back to central memory for that user. The program need not be concerned with the overwriting of data within a user area, since all user areas are protected by SCED.

The data to be sent to a terminal is written out from a buffer specified in the TERMOUT call. The buffer must be contained within the user area, so that SCED can transfer the data to be transmitted to an INTERCOM buffer while allowing the muj program to continue execution with another user. Although the program defines and uses a single user area, the program field length contains many buffers for users areas, as defined when the program is installed. These SCED buffers allow the program to have a single user area, yet allow terminal transmission to overlap another user execution.

Terminal input areas need not be defined in the user area because the input data is received in SCED buffers and held until the call for terminal input is issued by a specific user.

Record areas for data to be read from or written to files on mass storage should not appear in user areas. Record areas should be protected from overwriting by the use of interlocks discussed below.

Data cannot be preset in the user area. The program itself must put data in the user area during execution.

Interlocks

The many users executing a single muj program share the data areas as well as the Procedure Division statements. The item defined as the key-item for an indexed sequential file, for instance, is the same for all users accessing that file. To prevent the key entered by one user from being overwritten by the key entered by another user, SCED provides an interlock facility that protects a data area, even though the user who defined its contents has temporarily left execution.

An interlock is a SCED feature that permits one user to have exclusive access to statements that reference a data area such as a key-item or record area. All Procedure Division statements that lie between a GETINT call, which reserves the interlock, and a RETINT call, which releases the interlock, are said to fall within the range of the interlock. Only the user reserving the interlock can execute statements within its range. Other users are locked out of the interlocked statements, but are not locked out of the entire program. The program continues to be shared under SCED control.

The range of an interlock can encompass many statements and more than one SCED point. SCED points TERMIN and TERMOUT should not be used within an interlock, however, because such use would needlessly delay execution for other users.

Typically, an interlock is associated with each random access file in the program. When a user indicates that a record is to be read, for example, the interlock is reserved, the key-item is set, and a SEEK statement is executed for the record. An IOWAIT call to SCED immediately following the SEEK statement causes the user to leave the program and another user to execute in some other portion of the program. When control is returned to the user, a READ statement can move the requested data record to the data area associated with the key-item entered the last time the user was executing in the program. The program can continue by moving the record just retrieved to the TERMOUT buffer within the user area, issuing a RETINT call to release the interlock, and the next time in execution, issuing a TERMOUT call to send the record to the requesting terminal. If no other user is waiting for that interlock, the TERMOUT call is executed immediately. Otherwise, the next user requesting that interlock begins execution as soon as the interlock is released.

Each interlock is identified by name in the Working-Storage Section, and is initialized to a level 77 COMP-1 value, starting from 0.

SCED CALLS

SCED is called by ENTER statements. Parameters in the calls must be defined as elementary items in the Data Division. The calls are presented below, alphabetically.

CONNECT

The CONNECT call connects a user to the muj program. The call causes SCED to assign a user area and create appropriate entries in tables internal to

SCED. CONNECT also indicates the beginning of terminal session processing. It is executed once for each new user entering the program. The format of the CONNECT call is:

ENTER CONNECT.

DISCON

The DISCON call disconnects a user from the muj program and causes SCED to release all associated table entries or buffer areas for that user. It also indicates the end of the terminal session processing. DISCON is executed once for each user attached to the program. The call to DISCON must be followed either by the termination code or by a branch to the termination code. The format of the DISCON call is:

ENTER DISCON.

EXITMUJ

The EXITMUJ call terminates a muj program. It is used instead of the STOP RUN statement normally used to terminate a COBOL program that is not a muj program. The call to EXITMUJ must be the last statement in the program. SCED causes it to be executed when the last active user is disconnected from the program. The format of the EXITMUJ call is:

ENTER EXITMUJ.

GETINT

The GETINT call reserves an interlock and defines the beginning of a series of statements reserved exclusively for use of the user reserving the interlock. All other users are locked out of the interlocked portion of the muj program until the interlock is released by a call to RETINT. Interlocks should be used only when they are necessary to protect a data area that would be overwritten if another user was switched into the program. The format of the GETINT call is:

ENTER GETINT USING interlock-id

interlock-id — Name of an independent item whose value identifies an interlock. The value must be a level 77 COMP-1 integer from 0 to n-1, where n is the value set at installation time. The program should define a unique interlock for each random access file that can be used in the program.

INIT

The INIT call initializes the SCED processing. The call causes SCED to establish tables and buffers according to the information in the INIT call and the information installed in the SCED routines for a particular muj program. INIT must be the first call to SCED within the program. It is executed only once, when the program is first brought into execution. The format of the INIT call is:

ENTER INIT USING u-area, rest, abt.

u-area — Name of a data area allocated in the muj program that contains, at minimum, the buffer for terminal output. Each time a new user is switched into execution, SCED copies the information in this area to an internal SCED buffer, so that the data is not overwritten by information related to another user.

rest — Name of the paragraph containing the initialization procedures.

abt — Name of the paragraph containing a DISCON call to disconnect a user from the program. SCED returns control to this paragraph if a user is permanently disconnected from the muj program.

IOWAIT

The IOWAIT call relinquishes control of the program until a prior SEEK statement is completed. The call is used to minimize central processor use while file actions are in process. IOWAIT, if it is used at all, must be issued within the range of an interlock. SCED returns control to the program after the SEEK is completed. The format of the IOWAIT call is:

ENTER IOWAIT USING file-name.

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

RETINT

The RETINT call releases an interlock previously reserved by a call to GETINT, so that previously reserved sections of the program are available to other users. A release of an interlock that was not reserved causes a fatal error. The format of the RETINT call is:

ENTER RETINT.

TERMIN

The TERMIN call requests a terminal input operation. It causes SCED to switch a user out of execution until such time as input is available for the program to process. When terminal user input is available, SCED switches in the user, places the transmission from the user in the input area defined in the TERMIN call, and resumes program execution at the statement after the call. The first word of the area must contain a COMP-1 item specifying the number of following words allocated for the incoming data. If the data received from the terminal is larger than the input area, the data is truncated to area size.

TERMIN should not be issued within the range of an interlock because processing for other users would be unduly delayed. The format of the TERMIN call is:

ENTER TERMIN USING in-area.

in-area	Name of the terminal input area defined in the Data Division.

TERMOUT

The TERMOUT call requests that data be sent to the terminal. It causes SCED to switch out the user and the user area containing the output buffer, and to transmit the data to the terminal from the SCED buffer rather than directly from the program output buffer. TERMOUT should not be issued within the range of an interlock.

The muj program is responsible for formatting the data with the control characters required for terminal output. The first three fields of the terminal output area specified in the TERMOUT call must contain the specifications listed below.

buffer type	COMP-1 integer, one central memory word in length, corresponding to one of the buffer types defined to SCED when it was installed (see SCED Muj Installation later in this section).
length	COMP-1 integer, one central memory word in length, specifying the number of following data words to be transmitted to the terminal.
format character	One-character field set to an INTERCOM control character for interactive output.

Any number of lines can be transmitted with a single TERMOUT call. Each line should begin with a carriage control character and must end with a word containing zeros in bits 0-11. The format of the TERMOUT call is:

ENTER TERMOUT USING out-area.

out-area	Name of the terminal output area within the user area that contains two control words and data formatted for transmission to a terminal.

The INTERCOM reference manual contains additional information about the format control character.

SCED MUJ INSTALLATION

Installation of a muj program that calls SCED follows essentially the same procedure as installation of a muj program making direct calls to the muj subroutines. An example is given in section 4. SCED installation might involve an extra step, however, if the programmer chooses to alter any or all values in a group of SCED parameters. These parameters are altered by COMPASS macros added to the source statements of SCED by means of an UPDATE run before SCED assembly. (The exact location in the SCED source text where the macros are to be inserted is given in

the NOS/BE 1 installation handbook.) Default values are provided for all the parameters. The macros and their parameters are listed below.

MAXUSR num

Defines the maximum number of users that can be attached to the muj program at a given time. Subsequent users attempting to call the program will not be allowed to do so by INTERCOM.

num — is the maximum number of users; default 30.

NUMINT num

Defines the number of interlocks needed by the muj program. (Each random access file should be associated with an interlock.) Interlock identification numbers start from zero and extend to num-1.

num — is the number of interlocks; default 40.

USAREA num, len

Defines the number of user area buffers that are to remain in central memory. When SCED requires more user area buffers than the limit defined in USAREA, some buffers might be swapped out. Efficiency of the muj program execution increases in direct proportion to the number of user area buffers that are allowed to remain in central memory.

num — is the number of user area buffers to remain in central memory; default 2.

len — is the length, in words, of each user area buffer; default 214 words decimal.

DEFBUF

Specifies that output buffer definition is to take place. DEFBUF is required if OUTBUF macros are defined. The DEFBUF macro has no parameters.

OUTBUF num, len

Defines the number and length of output buffers. Allocation of these buffers is the responsibility of the muj program. The OUTBUF macro can occur as many times as desired; each occurrence defines a different type of output buffer. The type of output buffer is governed by the length specification. The first occurrence of the OUTBUF macro defines a group of buffers whose buffer type is 0 and whose length is given by len. Subsequent occurrences of OUTBUF define buffer types that are numbered consecutively. When an output buffer is referenced by the TERMOUT call to SCED, the first word of the output buffer must contain the buffer type as a COMP-1 item, and the second word must contain the length of the data to be output in words, also as a COMP-1 item. The length given in the second word of the output buffer is often less than the full output buffer length.

num — is the number of output buffers of this type.

len — is the length in words of each buffer of this type.

Two buffer types are defined by default:

```
OUTBUF  4,45    Buffer Type 0

OUTBUF  4,144   Buffer Type 1
```

MUJ INSTALLATION 4

A muj program cannot be submitted for execution as a normal user job; it must be installed in the system. Installation requires the use of the system EDITLIB utility to add the muj program either to the deadstart file or directly to the running system. In the latter case, operator intervention is required, and INTERCOM cannot be running at the time the system EDITLIB takes place.

Although the exact steps involved in installation may vary, the procedure outlined below can be used to install any muj program, whether it uses SCED or calls the muj subroutines directly. The basic steps to be followed in installing a muj program are listed below:

1. The user's muj program is compiled by FORTRAN Extended or COBOL, or assembled by COMPASS. (SCED muj programs are compiled by COBOL.)

2. The muj subroutines (and preprocessors, if used) are extracted by an UPDATE run from the program library; the program library must be a file containing the source code for all INTERCOM routines.

3. (SCED jobs only) SCED is extracted from the INTERCOM program library in an UPDATE run. The SCED parameters can be altered in the same run if desired.

4. The muj subroutines (and preprocessors, if used) are compiled by FORTRAN Extended.

5. (SCED jobs only) SCED is assembled by COMPASS.

6. The relocatable binary code produced in steps 4 and 5 is made absolute by the loader and written to a file. (It can be written in overlay form if requested by the user.)

7. The program written to a file in step 6 is added to a system library through the system EDITLIB utility.

8. A second UPDATE run adds entries to tables in the INTERCOM peripheral processor routines 1QP and 1CI. These entries enable INTERCOM to recognize the muj program name.

9. The updated peripheral processor routines are assembled by COMPASS and the object modules are written to a file.

10. A second system EDITLIB procedure replaces the versions of 1QP and 1CI in the system peripheral processor library with the versions created in step 9.

Some of the above steps can be consolidated or executed in a different order. They do not have to be executed in the same job that adds the muj program to the system library. For example, all the binaries can be generated in advance and written to a tape; the tape can then be used as input to the system EDITLIB procedure that adds the muj program to the system.

INSTALLATION EXAMPLE

Figure 4-1 shows a job structure that can be used to install a COBOL muj program that uses the COBOL preprocessing routines. The job assumes that the INTERCOM routines have been extracted from a release tape and cataloged with the permanent file name INTPL. The job performs a system EDITLIB to add the muj program to the running system.

The second statement compiles the COBOL muj program. The E parameter generates an OVERLAY (COBCODE,0,0) loader directive and specifies that the name used to call the program is MYMUJ. The relocatable object code is written to file MUJBIN.

Statement 3 attaches the permanent file that contains the INTERCOM routines in program library format.

Statement 4 extracts the muj subroutines (MUJSUBS) and the COBOL preprocessors (COBOLMUJ) from INTPL and writes them to the file COMPILE, following the directive on statement 19.

Statement 5 compiles the muj subroutines from the file COMPILE and writes the relocatable object code to MUJBIN.

Statement 6 compiles the preprocessors from the file COMPILE and writes the relocatable object code to MUJBIN.

Statement 8 creates an absolute binary of all the programs on MUJBIN and writes it as an overlay to the file COBCODE.

Statement 10 adds the user muj program to the system library NUCLEUS, following directives in statement 21 through 26.

Statement 12 extracts the peripheral processor routines 1CI and 1QP from INTPL, changes them according to directives following statement 28, and writes the changed tables to the file COMPILE.

Statements 13 and 14 assemble the changed versions of 1CI and 1QP and writes the object code on file LGO.

Statement 16 replaces the running system versions of 1CI and 1QP with the changed versions on LGO, following directives in statements 31 through 34.

The COBOL muj program should be inserted in the deck between statements 17 and 18.

Statement 19 contains directives for UPDATE called at statement 4. The deck named CWEOR6 occurs on INTPL between MUJSUBS and COBOLMUJ, and contains a *CWEOR directive to write an end-of-section on the COMPILE file. The exact order and names of decks on the old program library should be verified before an installation run is executed.

Statements 21 through 26 are system EDITLIB directives that add the COBOL muj program to the system library. Statement 23 must contain an FL parameter giving the field length required to execute the program and an AL parameter giving access level information. AL values depend on other INTERCOM installation parameters and should be chosen with the aid of a system analyst.

Statement 28 gives an UPDATE identifier that will be applied to directives between statements 28 and 29.

These directives must specify the location within the 1CI and 1QP tables where the name of the COBOL muj program is to be placed.

Statements 31 through 34 are system EDITLIB directives that add the new version of 1CI and 1QP to the running system.

VARIATIONS ON INSTALLATION PROCEDURES

The job structure in figure 4-1 is basically the same for muj programs written in FORTRAN or COMPASS. If the muj program uses SCED to interface with the muj subroutines, the SCED routines must be compiled in addition to the MUJSUBS routines.

FORTRAN VARIATIONS

If the muj program is written in FORTRAN Extended, make the following changes to the statement shown in figure 4-1.

Replace statement 2 with:

```
FTN,B=MUJBIN.
```

Ensure that the first statement (before the PROGRAM statement) in the FORTRAN program inserted between statements 17 and 18 is:

```
OVERLAY(COBCODE,0,0)
```

Replace statement 19 with a directive that extracts the muj subroutines and the FORTRAN preprocessors:

```
*C  MUJSUBS,CWEOR6,FTNMUJ
```

COMPASS VARIATIONS

If the muj program is written in COMPASS, make the following changes to the statements shown in figure 4-1.

Replace statement 2 with:

```
COMPASS,B=MUJBIN.
```

```
 1 JOB STATEMENT
 2 COBOL,B=MUJBIN,E=MYMUJ.
 3 ATTACH,OLDPL,INTPL,ID=WHOOS.
 4 UPDATE,Q,C=COMPILE.
 5 FTN,I=COMPILE,B=MUJBIN,L=0.
 6 FTN,I=COMPILE,B=MUJBIN,L=C.
 7 LOAD,MUJBIN.
 8 NOGO.
 9 REWIND,COBCODE.
10 EDITLIB,SYSTEM.
11 RETURN,COMPILE.
12 UPDATE,O,C=COMPILE.
13 COMPASS,I=COMPILE,B=LGO,L=3,
14 S=IPTEXT,S=PPTEXT,S=SCHTEXT.
15 REWIND,LGO.
16 EDITLIB(SYSTEM)
17 7/8/9

        COBOL MUJ PROGRAM

18 7/8/9
19 *C MUJSUBS,CWEOR6,COBOLMUJ
20 7/8/9
21 READY(SYSTEM,OLD)
22 LIBRARY(NUCLEUS,OLD)
23 REPLACE(*,COBCODE,FL=...)
24 FINISH.
25 COMPLETE.
26 ENDRUN.
27 7/8/9
28 *ID ANY
        UPDATE DIRECTIVES
29 *C 1CI,10P
30 7/8/9
31 READY(SYSTEM,OLD)
32 REPLACE(*,LGO)
33 COMPLETE.
34 ENDRUN.
35 6/7/8/9
```

Figure 4-1. Sample Deck for COBOL muj Program Installation

Include an LCC pseudo-instruction after the IDENT statement of the program inserted between statements 17 and 18 as:

```
LCC     OVERLAY(COBCODE,0,0)
```

Delete statement 6.

Replace statement 19 with a directive that compiles only the muj subroutines:

```
*C MUJSUBS
```

SCED VARIATIONS

If the muj program is written in COBOL and uses SCED instead of direct calls to the muj subroutines, make the following changes to the deck shown in figure 4-1.

Replace statement 6 with a call to assemble SCED:

```
COMPASS,I=COMPILE,B=MUJBIN,L=0.
```

Replace statement 19 with a directive that extracts the muj subroutines and SCED:

```
*C MUJSUBS,CWEOR6,SCED
```

If DUMMUJ is being used to debug a program calling SCED, the program need not be installed as part of the running system. Make the following changes to the deck shown in figure 4-1.

Replace statement 6 with a call to assemble DUMMUJ:

```
COMPASS,I=COMPILE,B=MUJBIN,L=0.
```

Replace statement 8 with:

```
EXECUTE.
```

Replace statement 19 with a directive that extracts the muj subroutines and DUMMUJ:

```
*C MUJSUBS,CWEOR6,DUMMUJ
```

Delete statements 9 through 16 and 20 through 34.

MUJ EXAMPLE 5

The muj program described in this section illustrates the basic structure of a muj program and some of the capabilities available. The purpose of the program, which is written in FORTRAN Extended and is called CONVERS, is to enable connected users to send messages to each other. Both sender and recipient of a message must be connected to CONVERS.

After connecting to the muj program by entering CONVERS at the terminal, the user sends as many one-line messages as desired by entering:

id, message

where id is the two-character use identification of the intended recipient (previously obtained via the INTERCOM SITUATE command), and message is 0-76 characters in any sequence.

When the user wishes to disconnect from the muj program, the characters END are entered at the beginning of a line and followed by a carriage return. The FORTRAN Extended source code for sample program CONVERS is shown in figure 5-1.

When the first INTERCOM user calls CONVERS, the program is loaded from the system library and execution begins with the first executable statement, the call to INMUJ (statement 7). The call to INMUJ is executed once each time CONVERS is loaded; it is executed only for the first user. When subsequent users call CONVERS, they are put into a scheduling queue by INTERCOM. Eventually each of these users is given to CONVERS as a new user on return from a call to USER (statement 9). The program contains only one call to USER; the program repeatedly branches on this call to process user requests and receive new users. The parameter STATE is then examined to determine the nature of the user request. Three possible user requests can be made:

The user asks to be connected to the muj program by entering CONVERS. When the user is sent by USER to CONVERS as NEWU, the parameter equals 5. In this simple muj program, no special action is taken at that time; other programs might perform initialization processing, such as setting up tables for each user.

The user requests that a message be sent to another terminal. In this case, state=1, since USER does not schedule the user until the input from the terminal is ready. CONVERS determines the recipient from the two-character user identification and formats the message. TIO is then called to send the message to the recipient.

The user asks to exit the muj program. Exit can be achieved in one of two ways: either the user types END, and is returned by USER with state=1, or the user enters a break of %A, and bit 59 of the state parameter is set. It is also possible that the user has been automatically disconnected by INTERCOM; in this case, state=8. In all of these cases, CONVERS requests user disconnection by returning the user to USER with actn=-5.

If any state other than those described above is detected, CONVERS simply returns the user to the scheduling queue.

```
      OVERLAY(MUJABS,0,0)
      PROGRAM CONVERS
      DIMENSION MUJTBL(54),UAREA(60)
      INTEGER STATE,OLDU,ACTN,UAREA
      DATA IAREA,STATE,NEWU,OLDU,ACTN/5*0/
C     INITIALIZE (CALLED ONLY ONCE FOR EACH LOAD)
      CALL INMUJ(MUJTBL,0,6,6,6,10,UAREA)
C     MAIN LOOP -- JUST KEEP CALLING USER TO PROCESS REQUESTS
100   CALL USER(OLDU,ACTN,STATE,NEWU,IAREA)
      OLDU=NEWU
      NEWU=0
C     IF BREAK HAS OCCURRED --
      IF(STATE.LT.0) GO TO 108
      STATE=STATE.AND.777777B
      GO TO (101,100,100,100,105,100,100,108) STATE
C     WE  HAVE A MESSAGE
101   NW=8
      CALL TIO (OLDU,3,UAREA(IAREA+2),NW)
C     CHECK FOR TERMINATION
      IF((UAREA(IAREA+2).AND.MASK(18)).EQ.3LEND) GO TO 108
C     EXTRACT ID OF RECIPIENT
      ID=UAREA(IAREA+2).AND.MASK(12)
      UAREA(IAREA)=5H FROM
      UAREA(IAREA+1)=UAREA(IAREA+2).AND.MASK(12).OR.8R      ::
C     REPLACE ≠ID,≠ WITH BLANKS
      UAREA(IAREA+2)=UAREA(IAREA+2).AND.COMPL(MASK(18)).OR.3L
C     SEND MESSAGE
      CALL TIO (ID,2,UAREA(IAREA), NW + 2)
      ACTN=-1
      GO TO 100
C     GET NEXT USER
105   ACTN=-1
      GO TO 100
C     DISCONNECT USER
108   ACTN=-5
      GO TO 100
      END
```

Figure 5-1. Sample Program CONVERS

STANDARD 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.

STANDARD 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	072	6	6	41	6	06	6	066
A	A	01	12-1	61	12-1	101	7	7	42	7	07	7	067
B	B	02	12-2	62	12-2	102	8	8	43	8	10	8	070
C	C	03	12-3	63	12-3	103	9	9	44	9	11	9	071
D	D	04	12-4	64	12-4	104	+	+	45	12	60	12-8-6	053
E	E	05	12-5	65	12-5	105	–	–	46	11	40	11	055
F	F	06	12-6	66	12-6	106	*	*	47	11-8-4	54	11-8-4	052
G	G	07	12-7	67	12-7	107	/	/	50	0-1	21	0-1	057
H	H	10	12-8	70	12-8	110	(	(	51	0-8-4	34	12-8-5	050
I	I	11	12-9	71	12-9	111	)	)	52	12-8-4	74	11-8-5	051
J	J	12	11-1	41	11-1	112	$	$	53	11-8-3	53	11-8-3	044
K	K	13	11-2	42	11-2	113	=	=	54	8-3	13	8-6	075
L	L	14	11-3	43	11-3	114	blank	blank	55	no punch	20	no punch	040
M	M	15	11-4	44	11-4	115	, (comma)	, (comma)	56	0-8-3	33	0-8-3	054
N	N	16	11-5	45	11-5	116	. (period)	. (period)	57	12-8-3	73	12-8-3	056
O	O	17	11-6	46	11-6	117	≡	#	60	0-8-6	36	8-3	043
P	P	20	11-7	47	11-7	120	[	[	61	8-7	17	12-8-2	133
Q	Q	21	11-8	50	11-8	121	]	]	62	0-8-2	32	11-8-2	135
R	R	22	11-9	51	11-9	122	%	%	63††	8-6	16	0-8-4	045
S	S	23	0-2	22	0-2	123	≠	" (quote)	64	8-4	14	8-7	042
T	T	24	0-3	23	0-3	124	→	_ (underline)	65	0-8-5	35	0-8-5	137
U	U	25	0-4	24	0-4	125	∨	!	66	11-0 or	52	12-8-7 or	041
V	V	26	0-5	25	0-5	126				11-8-2†††		11-0†††	
W	W	27	0-6	26	0-6	127	∧	&	67	0-8-7	37	12	046
X	X	30	0-7	27	0-7	130	↑	' (apostrophe)	70	11-8-5	55	8-5	047
Y	Y	31	0-8	30	0-8	131	↓	?	71	11-8-6	56	0-8-7	077
Z	Z	32	0-9	31	0-9	132	<	<	72	12-0 or	72	12-8-4 or	074
0	0	33	0	12	0	060				12-8-2†††		12-0†††	
1	1	34	1	01	1	061	>	>	73	11-8-7	57	0-8-6	076
2	2	35	2	02	2	062	≤	@	74	8-5	15	8-4	100
3	3	36	3	03	3	063	≥	\	75	12-8-5	75	0-8-2	134
4	4	37	4	04	4	064	¬	~(circumflex)	76	12-8-6	76	11-8-7	136
5	5	40	5	05	5	065	; (semicolon)	; (semicolon)	77	12-8-7	77	11-8-6	073

†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.

DIAGNOSTIC MESSAGES B

The diagnostic messages generated by multi-user job processing, as described in this manual, are listed by error number. Each message is described by the significance of the message, the action to be taken, and the issuing routine.

The header message MUJ SYSTEM ERROR xx, where xx is the error code number, is routed to both the system dayfile and each user's terminal whenever an error occurs. Values of xx that are less than 50 indicate error conditions encountered by the system muj subroutines; values of 50 or greater indicate errors detected by SCED.

TABLE B-1. DIAGNOSTIC MESSAGES

Error Code	Significance	Action	Issuing Routine
0	A system error, such as operator drop, a mode error, or a peripheral processor abort, occurred.	Notify systems analyst.	REKOVER
1	The user area was lost internally.	Notify systems analyst.	ADJUAR/ SRCUAR
2	Not used.		
3	Bit KWCON was set for this value of MMACT. (FATAL)	Notify systems analyst.	SERVICE
4	An error from CIO occurred on the last user area swap.	Notify systems analyst.	SWAPOK
5	An illegal CIO function code was received on the last user area swap. (FATAL)	Notify systems analyst.	SWAPOK
6	The user area was lost on swapout. (FATAL)	Notify systems analyst.	SEGSWPD
7	A CIO error code occurred during terminal output.	Notify systems analyst.	SERVICE
8	Not used.		
9	The muj program returned a user area not currently assigned to it. (FATAL)	Programmer action required.	USER
10	An invalid ACTN code was sent by the muj program. (FATAL)	Programmer action required.	USER
11	Invalid information was received from 1QP.	Notify systems analyst.	USER

TABLE B-1. DIAGNOSTIC MESSAGES (Continued)

Error Code	Significance	Action	Issuing Routine
12	New user already exists in MUJTERM.	Notify systems analyst.	USER
13	A non-ready user was marked ready.	Notify systems analyst.	USER
14	A logical unit number was specified in the call to USERFO, but corresponding file was not declared on the muj PROGRAM card. (FATAL)	Programmer action required.	NSCFET
15	Not used.		
16	The muj program is returning a user not assigned to it. (FATAL)	Programmer action required.	USER
17	The user's files cannot be returned when the user leaves the muj program.	Notify systems analyst.	USER
18–42	Not used.		
43	Binary-to-decimal conversion error from MSYSERR.	Bad error code passed to MSYSERR.	NUMCHAR
44–49	Not used.		
50	INIT was called out of sequence.	INIT must be first call to SCED in muj program.	SCED
51	CONNECT was called out of sequence.	CONNECT call must be preceded by a call to INIT.	SCED
52	Terminal session section function was called out of sequence.	Calls to INIT and CONNECT must precede calls to terminal session functions.	SCED
53	DISCON was called out of sequence.	DISCON call must be preceded by a call to CONNECT.	SCED
54	EXITMUJ was called out of sequence.	EXITMUJ call must be preceded by call to DISCON.	SCED
55–59	Not used.		

TABLE B-1. DIAGNOSTIC MESSAGES (Continued)

Error Code	Significance	Action	Issuing Routine
60	Interlock number specified in GETINT call is outside of range specified during installation.	Programmer action required.	SCED
61	User attempted to reserve an interlock already reserved for that user.	Programmer action required.	SCED
62	Interlock number specified in RETINT call is outside of range specified during installation.	Programmer action required.	SCED
63	User attempted to release an unreserved interlock.	Programmer action required.	SCED
64	User attempted to disconnect from muj program while an interlock is still reserved for the user.	Programmer action required.	SCED
65-69	Not used.		
70	Buffer type is outside range specified during installation.	Programmer action required.	SCED
71	Word count is greater than size specified for buffer type requested.	Programmer action required.	SCED
72	TIO error, usually caused by incorrectly formatted terminal output.	Programmer action required.	SCED
73-79	Not used.		
80	TIO error occurred on read.	Notify systems analyst.	SCED
81	No free user ordinal is available for new user. (Maximum number of users specified during installation exceeded.)	Notify systems analyst.	SCED
82	ID of new user is already in use.	Programmer action required.	SCED
83	Muj subroutines returned ID unknown to SCED.	Notify systems analyst.	SCED
84	Muj subroutines status code is not handled by SCED.	Notify systems analyst.	SCED

INDEX

COMMENT SHEET

MANUAL TITLE CDC INTERCOM Version 4 Multi-User Job Capability Reference Manual

PUBLICATION NO. 60494700 REVISION B

FROM: NAME:

BUSINESS ADDRESS:

CUT ALONG LINE

AA3419 REV. 7/75 PRINTED IN U.S.A.

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
ARH219
4201 North Lexington Avenue
Saint Paul, Minnesota 55112

CUT ALONG LINE

FOLD

FOLD

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

LITHO IN U.S.A.

CONTROL DATA CORPORATION