Price: $1.75

REAL-TIME BATCH MONITOR OPERATIONS MANUAL

for

XDS SIGMA 5/7 COMPUTERS

FIRST EDITION

90 16 47A

February 1970

XDS

Xerox Data Systems/701 South Aviation Boulevard/El Segundo, California 90245

RELATED PUBLICATIONS

Title	Publication No.
XDS Sigma 5 Computer Reference Manual	90 09 59
XDS Sigma 7 Computer Reference Manual	90 09 50
XDS Sigma 5/7 Real-Time Batch Processing Reference Manual	90 15 81
XDS Sigma 5/7 Real-Time Batch Monitor Users Guide	90 16 47
XDS Sigma 5/7 Real-Time Batch Monitor Technical Manual	90 17 00
XDS Sigma 5/7 Mathematical Routines Technical Manual	90 09 06
XDS Sigma 5/7 Symbol and Meta-Symbol Reference Manual	90 09 52
XDS Sigma 5/7 Macro-Symbol Reference Manual	90 15 78
XDS Sigma 5/7 SL-1 Reference Manual	90 16 76
XDS Sigma 5/7 FORTRAN IV-H Reference Manual	90 09 66
XDS Sigma 5/7 FORTRAN IV-H Operations Manual	90 11 44
XDS Sigma 5/7 FORTRAN IV-H Library/Run-Time Technical Manual	90 11 38
XDS Glossary of Computer Terminology	90 09 57

CONTENTS

ILLUSTRATIONS

TABLES

DEFINITION OF TERMS

active foreground program: a foreground program is active if it is resident in memory, connected to interrupts, or in the process of being entered into the system via a !RUN control command.

background area: that area of core storage allocated to batch processing. This area may be checkpointed for use by foreground programs.

background program: any program executed under Monitor control in the background area with no external interrupts active. These programs are entered through the batch processing input stream.

binary input: input from the device to which the BI (binary input) operational label is assigned.

centrally connected interrupt: an interrupt that is connected to a Monitor interrupt routine which first saves the environment of the system and then switches the environment to that of the task that gets control when the interrupt occurs.

checkpointed job: a partially processed background job that has been saved in secondary storage along with all registers and other "environment" so that the job can be restarted.

control command: any control message other than a key-in. A control command may be input via any device to which the system command input function has been assigned (normally a card reader).

control message: any message received by the Monitor that is either a control command or a control key-in (see "key-in").

Data Control Block (DCB): a table in the executing program that contains the information used by the Monitor in the performance of an I/O operation.

dedicated memory: core memory locations reserved by the Monitor for special purposes, such as traps, interrupts, and real-time programs.

directly connected interrupt: an interrupt which, when it occurs, causes control to go directly to the task. E.g., execution of the XPSD instruction in the interrupt location gives control to the task rather than first going to a Monitor interrupt routine.

end record: the last record to be loaded, in an object module or load module.

execution location: a value replacing the origin of a relocatable program, to change the address at which program loading is to begin.

foreground area: that portion of memory dedicated specifically for foreground programs.

foreground program: a load module that contains one or more foreground tasks.

foreground task: a body of procedural code that is associated with (connected to) a particular interrupt and that is executed when the interrupt occurs.

Function Parameter Table (FPT): a table through which a user's program communicates with a Monitor function (such as an I/O function).

GO file: a temporary disc file of relocatable object modules formed by a processor.

idle state: the state of the Monitor when it is first loaded into core memory or after encountering a !FIN control command. The idle state is ended by means of a C key-in.

installation control command: any control command used during System Generation to direct the formatting of a Monitor system.

key-in: information entered by the operator via a keyboard.

keyword: a word, consisting of from 1 to 8 characters, that identifies a particular operand used in a control command.

library input: input from the device to which the LI (library input) operational label is assigned.

load map: a listing of significant information pertaining to the storage locations used by a program.

load module: an executable program formed by using relocatable object modules and/or library object modules as source information.

logical device: a peripheral device that is represented in a program by an operational label (e.g., BI or BO) rather than by a specific physical device name.

Monitor: a program that supervises the processing, loading, and execution of other programs.

object deck: a card deck comprising one or more object modules and control commands.

object language: the standard binary language in which the output of a compiler or assembler is expressed.

object module: the series of records containing the load information pertaining to a single program or subprogram. Object modules serve as input to the Overlay Loader.

operational label: a symbolic name used to identify a logical system device.

option: an elective operand in a control command or procedure call, or an elective parameter in a Function Parameter Table.

Overlay Loader: a processor that loads and links elements of overlay programs.

overlay program: a segmented program in which the segment currently being executed may overlay the core storage area occupied by a previously executed segment.

physical device: a peripheral device that is referred to by a "name" specifying the device type, I/O channel, and device number (also see "logical device").

postmortem dump: a listing of the contents of a specified area of core memory, usually following the abortive execution of a program.

Relocatable Object Module: a program, or subprogram, generated by a processor such as Macro-Symbol, FORTRAN, etc. (in XDS Sigma 5/7 object language).

resident program: a program loaded into core each time the system is booted in.

ROM: Relocatable Object Module.

secondary storage: any rapid-access storage medium other than core memory (e.g., RAD).

segment loader: a Monitor routine that loads overlay segments from disc storage at execution time.

source deck: a card deck comprising a complete program or subprogram, in symbolic EBCDIC format.

source language: a language used to prepare a source program (and therefrom a source deck) suitable for processing by an assembler or compiler.

standard control section: a control section whose length is not known by a 1-pass processor until all the load information for that section has been generated.

symbolic input: input from the device to which the SI (symbolic input) operational label is assigned.

symbolic name: an identifier that is associated with some particular source program statement or item so that symbolic references may be made to it even though its value may be subject to redefinition.

system library: a group of standard routines in object-language format, any of which may be included in a program being created.

Task Control Block (TCB): a table of program control information built by the relocating loader when a load module is formed. The TCB is part of the load module and contains a temp stack and the data required to allow reentry of library routines during program execution. The TCB is program associated and not task associated.

1. INTRODUCTION

The XDS Sigma 5/7 Real-Time Batch Monitor (RBM) is the major control element of the operating system described in this manual. The total operating system includes the Monitor, Overlay Loader, RAD Editor, language processors, and user real-time and batch programs.

The content of this manual is operator-oriented in that it is specifically directed toward Monitor/operator communications, procedures, control command formats, and device considerations necessary to maintain the system and process program inputs under Monitor control. A comprehensive discussion of the internal functions of the Monitor and its associated components is given in the XDS Sigma 5/7 Real-Time Batch Monitor Reference Manual (Publication No. 90 15 81).

The RBM system divides core memory and Rapid Access Data file (RAD) utilization into two distinct areas: foreground (privileged) areas and background areas. The purpose of this division is to guarantee real-time tasks adequate memory and protection in the execution of highly critical processes, and at the same time, offer an efficient method for using up idle CPU time when real-time processes are not busy. Allocation of both areas is performed at System Generation time (SYSGEN) in accordance with the needs of the local installation.

Typically, the operator's responsibilities are directed toward two distinct applications of program execution: real-time processing and batch (background) processing.

REAL-TIME PROGRAMS

Real-time programs are connected to hardware interrupts that receive signals from external sources or real-time clocks to trigger execution and respond to these external events within microseconds.

Some real-time programs are loaded and initialized at System Boot time. Others can be run and released by the operator. The first method is used when the real-time process normally remains unchanged and is constantly operative. Typical examples would be a satellite tracking system or the control element of an automated plant or factory. The second approach is sometimes used when real-time operations are executed periodically or irregularly. An example would be a test procedure in an experimental laboratory. In this case, the operator must access the protected portion of memory through an FG key-in to load and initialize the program (see Table 2 for key-in descriptions).

BACKGROUND PROGRAMS

Background jobs are assembled or compiled and executed in the background area of core memory. The RBM operating system is designed to allow background programs to use up all available CPU time when the real time processes are not operative, thus giving greater economy to the system.

In contrast to real-time tasks whose priority sequence and function is controlled by external hardware interrupt or operator key-in, background programs are executed in a serial fashion and their sequence is controlled by control commands inserted in the job stack.

Background compilations or assemblies are initially loaded from some peripheral device (generally a card reader) onto a file in the Background Programs area of the RAD and executed in background core storage in serial fashion.

SOFTWARE ENVIRONMENT

The software modules and files with which the operator will find himself directly or indirectly involved are

- Monitor
- Job Control Processor
- Overlay Loader
- OV file
- Permanent RAD Files
- RAD Editor
- System Processors
- GO File
- User Programs (foreground and background)

MONITOR

The Monitor is the primary control element of the operating system and functions as a supervisor that coordinates and controls a continuous series of foreground and background jobs. It is also the two-way communications link between the operator and the total system. The operator can direct the system to change the status of an I/O device and alter system operation. The Monitor communicates with the operator via the operator's console (OC device) and in some instances, the listing log (LL device, which is generally a line printer) with error messages, status messages, or requests for operator action such as readying a magnetic tape unit.

Other services the Monitor performs includes preserving the relative priority of real-time tasks, protecting the foreground core and secondary storage from background interference, monitoring available CPU time for background use when real-time interrupts are idle, and providing general I/O services for all tasks.

Parts of the Monitor are permanently resident in core storage so that it can respond immediately to a request for service from a real-time task. Response time to such tasks is highly critical and is measured in microseconds. Less critical portions of the Monitor such as the Job Control Processor are permanently stored on the Systems area of the RAD and brought into core as needed.

JOB CONTROL PROCESSOR

The Job Control Processor (JCP) reads and processes control commands input from the Control Command (C) input device. Whenever the JCP encounters a request in the job stack to execute a processor (such as the FORTRAN compiler), or a user program, it causes the processor or program to be loaded into core from the RAD and the Monitor to relinquish control.

The JCP outputs error and status messages to the operator on both the LL and OC devices (unless both are assigned to the same device) in contrast to other Monitor messages that are output only on the OC device. The only significance the JCP has to the operator is that at least one standard status message to the operator refers to the JCP; he may also wonder why some messages are output only on OC and others are output on both OC and LL. Generally, messages output on LL are for programmer reference. All messages of interest to the operator are output on the OC device.

OVERLAY LOADER

The Overlay Loader creates programs in overlay (segmented) form. That is, the Loader converts a program in object module form into an absolute core image version called a load module that is segmented for later execution. Segmenting permits programmers to create programs much larger than the available core size.

Like any other processor, the Loader is called in for execution by a control command in the user's job. The mnemonics for the control commands used by the Overlay Loader are similar to those used by the Monitor but the format of the commands is slightly different (see the Overlay Loader chapter later in this manual). When the Loader has control, it communicates with the operator through its own status and error messages.

RAD FILES

Since the entire RBM system is RAD oriented, every job run by the operator will directly or indirectly involve the use, modification, allocation, or release of permanent RAD files. The files of greatest interest to the operator and the mnemonics used to reference them on control commands are as follows:

- Foreground Programs area (FP) contains a collection of foreground programs and optional User Libraries and Public Libraries. Referenced library routines are included in the user load modules at "Load" time. Public Libraries are a group of routines shared by a number of programs and are called into core for execution only when referenced.

- System Programs area (SP) contains the Monitor and the set of language processors used by the local installation, such as Macro-Symbol and FORTRAN IV-H. The area also contains the System Library (i.e., FORTRAN IV-H Library/Run-time), RAD Editor, and Overlay Loader. All translators are called by user jobs to execute in the background core space.

- Background Programs area (BP) contains the set of user operational programs that execute in the background and a background User Library if desired.

- Data areas (D1 through DF) are divided into foreground and background data areas, and are used for storing data. Foreground programs cannot write into files in background data areas or vice versa; however, either type of program can freely read from both areas.

- Background Temp area (BT) contains temporary (scratch) files (X1 through Xn where n is a SYSGEN parameter) used by background programs for intermediate storage in processing. Their use is identical to scratch tapes on magnetic tape units, and files can be rewound or searched. Note that temp files are erased at the end of each job step by having their pointers reset, unless a SAVE command is present within a job step. Temp files are automatically reset when a new JOB command is encountered in a job stack regardless of SAVE commands, and there is no way to save data on temp files from one job to another.

  The GO and OV files are also in the BT area and are special cases. The GO file contains Relocatable Object Modules (ROMs) formed by a processor if the GO option is specified. The OV file contains the executable program formed by the Overlay Loader if a program file name was not specified at load time.

RAD EDITOR

The RAD Editor controls RAD allocation for areas containing permanent RAD files and performs utility functions for all areas.

The operator will encounter a number of jobs involving RAD file manipulation via the RAD Editor, including: allotting RAD files, building RAD files, dumping RAD areas or files on user request, copying object modules from libraries, loading new programs into user or system libraries, inhibiting bad RAD tracks, etc.

RAD file manipulation via the RAD Editor is governed by control commands that have a format similar to Monitor control commands. When the RAD Editor has control, it communicates with the operator through its own error and status messages.

SYSTEM PROCESSORS

The following language processors are available under RBM, and any or all of them may be incorporated in the local system:

FORTRAN IV-H

SL-1

Symbol

Macro-Symbol

The selected processors are permanently stored in the System Programs area on the RAD and called into core storage to assemble or compile a user's source program through a control command in a user's source deck. The Monitor relinquishes control to the requested processor until the job step is completed.

2. OPERATOR/SYSTEM INTERFACE

Communication between the operator and the system takes place through operator key-ins (solicited or unsolicited), and Monitor printouts. In addition to job status messages, Monitor type-outs on the keyboard/printer inform the operator of various abnormal or error conditions affecting system operation. All Monitor messages to the operator are preceded by two exclamation marks (!!).

UNSOLICITED KEY-INS

Unsolicited key-ins provide the operator with a means of controlling a background job, or the loading or releasing of foreground programs. Any control the operator can exercise over the foreground is provided through key-ins, so that foreground control is independent of the background job stack.

Depressing the INTERRUPT switch on the Control Panel initiates a key-in.[†] The types of key-ins available are listed in Table 2. When all foreground tasks are inactive, the Monitor responds with the message

```
!! KEY-IN
```

on the operator's console. The operator then keys in the symbol and data he wishes to transmit to the system. Two conventions apply to all key-ins:

1. End each message by depressing the NL (New Line) key.
2. Delete a message by depressing the EOM (End of Message) key before depressing the NL key.

Use of an exclamation mark to precede an operator key-in is optional. Spaces (blanks) to separate fields can be used as desired.

DEVICE NAMES

Some key-ins require a device name. Devices are identified by type, channel, and device number. Device types and their associated symbols are listed in Table 1. The eight available channels are represented by a single letter, from A to H. Device numbers are given in two-digit hexadecimal code, from 00 to 7F. Alternatively, device controllers and their associated devices may be identified by a single hexadecimal digit from 8 to F for the device controllers, followed by a single hexadecimal digit from 0 through F for the associated device.

[†]If the foreground is busy when a key-in is initiated, the Monitor will not respond until all foreground interrupts are satisifed.

Examples:

CRA03	Card reader, Channel A, device 3
7TAE1	7-track magnetic tape, Channel A, device controller E, device 1.

Table 1. I/O Device Type Codes

Mnemonic	Device Type
TY	Typewriter
LP	Line Printer
CR	Card Reader
CP	Card Punch
9T	9-track magnetic tape
7T	7-track magnetic tape
PP	Paper tape punch
PR	Paper tape reader
DC	Magnetic disc or XDS RAD
NO	Not a standard device. Used as a special purpose device for direct I/O execution (IOEX).
PL	Plotter

OPERATIONAL LABELS

An operational label identifies a logical device type so that it can be assigned to a specific I/O device. For example, the operational label BI stands for binary input. This type of data is usually assigned to a card or tape reader. Operational labels are assigned at system generation time but may be changed by a STDLB (for standard label) key-in. At the beginning of each job, the operational labels are reassigned to their initial assignment. System operational labels are listed below.

Label	Meaning
BI	Binary input
CI	Compressed input (Macro-Symbol)
SI	Source input (symbolic)
C	Control command input[†]
BO	Binary output

[†]Both the temporary and permanent assignments of this label can be changed through STDLB key-in or control command unless "C" is assigned to zero. In this event, STDLB will only change the temporary label.

Label	Meaning
DO	Diagnostic output
LO	Listing output
CO	Compressed output (Macro-Symbol)
LL	Listing log (control commands and other system messages)
OC	Operator's console data
SO	Symbolic output (SL-1)
PL	Plotter output

STANDARD KEY-INS

The analysis and subsequent action from an unsolicited key-in is performed at the RBM Control Task priority level. When the RBM Control Task becomes the highest priority in the system (that is, when all foreground tasks are inactive) the message

```
!!KEY-IN
```

will be output on the OC device to inform the operator that the system can now accept a key-in.

Note that if the typewriter is busy at the time of the Control Panel Interrupt (i.e., waiting for an input to complete), the operator must complete the input before the !!KEY-IN type-out can occur. All legal key-ins are itemized in Table 2.

Table 2. Operator Key-Ins

Key-In	Purpose
X	Abort current background job. Message on OC and LL will show last location executed.
C	Continue processing in the background. If the background was in a wait or idle state, the system leaves that state and proceeds.
COC	Job was halted because of error in control command. Continue from OC with correct control command (after depressing the Control Panel interrupt key).
DT mo,day,yr,hr,min	Input of current date and time. Example: DT 8,17,69,22,30
RUN name	Load and execute a foreground program. The name of the foreground file to be loaded must be input.
TY	Transfer control from the C device to OC (typewriter).
SY	Override RAD software protection to allow a background program to write in any RAD file. The SY key-in is cleared when the next !JOB command is encountered.
FG	Permit loading of foreground program from background job stack for execution via a !RUN control command.
CC	Retransfer control back to the C device from OC.
RLS name	Release foreground program designated in the (file) name.
STDLB label [,area] ,name	Change permanent operational label assignment. The "label" identifies the op label. The "area" specifies a RAD area if the new oplb is assigned to a RAD file; the "name" specifies a physical device or file to which the oplb is to be assigned. The new assignment stays in effect until the system is rebooted or the assignment changed by another key-in.
INTLB label, location (hex) of interrupt	Change assignment of interrupt labels.

Table 2. Operator Key-Ins (cont.)

Key-In	Purpose
CINT $\left\{\begin{matrix}\text{location}\\ \text{label}\end{matrix}\right\}, \left\{\begin{matrix}\text{D}\\ \text{A}\\ \text{T}\end{matrix}\right\}$	Disarm, arm and enable, or trigger specified interrupt. The "location" specifies the hex address of the interrupt; "label" specifies an interrupt label; "D" is used to disarm specified interrupt; "A" is used to arm and enable; "T" is used to arm, enable, and trigger the interrupt.
FMEM [n]	Change foreground core memory allocation. The "n" specifies number of pages to be allocated to the foreground. Reallocation takes place after current background job step is completed. If the foreground is not free, the alarm !!FGD AREA ACTIVE is output. If "n" is not specified, the foreground is reset to the allocation specified at SYSGEN.
$\left\{\begin{matrix}\text{DM}\\ \text{DF}\\ \text{DB}\end{matrix}\right\}$ [from,to]	Dump Monitor (DM), Dump Foreground (DF), or Dump Background (DB) areas of memory. If "from, to" is absent, the entire currently defined area will be dumped; if present, the first word address in hex and last word address in hex of the selected area are defined.
DED yyndd $\left\{\begin{matrix}\text{F}\\ \text{X}\end{matrix}\right\}$ [,I]	Dedicate a device, and its controller and, optionally, all other devices on the same IOP to the foreground or to IOEX. The "yyndd" is the name of the device to be dedicated; the "F" specifies dedication to foreground; "X" specifies dedication to IOEX; "I" specifies all devices on IOPn are to be dedicated.
UND yyndd $\left\{\begin{matrix}\text{F}\\ \text{X}\end{matrix}\right\}$ [,I]	Undedicate a device or IOP previously dedicated through DED. All parameters present in the DED previously used must be present in the UND key-in.

COMBINED KEY-INS

To expedite operator key-ins, the following combinations of key-ins are recognized:

Combined Form	Result
FGC	Executes FG and C key-ins.
SYC	Executes SY and C key-ins.
SFC, FSC	Executes FG,SY, and C key-ins.
TYC	Executes TY and C key-ins.

CORRECTING A KEY-IN

If an operator key-in is not recognized by the system as valid input, the message

!! KEY ERR

is output on OC. The operator should retype the correct key-in.

DEVICE CONTROL

If the Monitor encounters an abnormal condition during an I/O operation, a pertinent message to the operator is output on the OC device. Such a message is of the form

!! name message

where

name is the physical device name (see ASSIGN control command).

message is the message string informing the operator of the specific condition that has been detected; for example:

ERROR (error was detected on operation)

or

MANUAL (device not ready)

Monitor I/O messages are discussed below, grouped according to the type of device to which they apply.

After correcting the abnormal conditions, the operator responds by means of a key-in. The format for an I/O key-in is

name a

where

name is the physical device name of the device involved in the I/O operation.

a specifies a Monitor-action character (see Table 3).

Table 3. Monitor Actions

a	Monitor Action
C	Continue "as is".
E	Inform the user program of the error and transmit record "as is".
R	Repeat the I/O operation.

CARD READER

If the card reader fails to read properly, or if a validity error occurs, the Monitor outputs the message

```
!! CRndd ERROR
```

on the OC device. After correcting the condition, the operator responds with an I/O key-in message. The action character selected (see Table 3) depends on the circumstances.

If a feed check error or a power failure occurs, the Monitor outputs the message

```
!! CRndd ERROR
```

or

```
!! CRndd TIMED OUT
```

on the OC device,[t] depending on where in the cycle the error took place. If the card in the hopper is damaged, the operator replaces it with a duplicate; then, in either case he presses the RESET and START buttons on the card reader, and responds to the Monitor with the key-in

```
CRndd R
```

If the card stacker is full, the hopper is empty, or the device is in the manual mode, the Monitor outputs the message

```
!! CRndd MANUAL
```

on the OC device. The operator corrects the condition and then presses the START button on the card reader.

CARD PUNCH

Instead of outputting an error message when a punch error is detected, the I/O handler attempts to punch a card x times (x = NRT, a DCB parameter specified by the user)[tt] before outputting the message.

```
!! CPndd ERROR
```

[t]This message also occurs if CRndd does not respond to SIO within 5 seconds.

[tt]For a comprehensive discussion of this parameter, see Chapter 4, DCB Creation, in the Sigma 5/7 Real-Time Batch Monitor Reference Manual.

on the OC device. The above message indicates that the card punch is not functioning properly, and the operator should reevaluate the job stack based on this knowledge. Improperly punched cards are routed to an alternate stacker.

If the input hopper is empty, the stacker is full, chip box is full, or device is in manual mode, the Monitor outputs the message

```
!! CPndd MANUAL
```

on the OC device. The operator corrects the condition and presses the START button on the card punch.

If a power failure or a feed check error occurs, the Monitor outputs the message

```
!! CPndd ERROR
```

or

```
!! CPndd TIMED OUT[t]
```

on the OC device, depending on where in the cycle the error took place. If the card in the hopper is damaged, the operator removes it; then, in either case, he presses the RESET and START buttons on the card punch and responds to the Monitor with the key-in

```
CPndd R
```

PRINTER

When an irrecoverable print error is detected, the Monitor outputs the message

```
!! LPndd ERROR
```

on the OC device. The I/O handler attempts to print a line x times (x = NRT, a DCB variable specified by the I/O user) before outputting the above message. The operator's response after correcting the condition depends on the specific device and circumstances.

If the printer is out of paper, the carriage is inoperative, or the device is in the manual mode, the Monitor outputs the message

```
!! LPndd MANUAL
```

on the OC device. The operator corrects the condition and presses the START button on the line printer.

If the line printer power is off, the Monitor outputs the message

```
!! LPndd UNRECOG
```

[t]This message also occurs if CPndd does not respond to SIO within 5 seconds.

on the OC device. The operator should correct the condition and respond with the key-in

LPndd R

PAPER TAPE READER

If an error occurs during the reading of paper tape, the Monitor outputs the message

!! PRndd ERROR

on the OC device. After correcting the condition, the operator responds with an I/O key-in message. The action character selected (see Table 3) depends on the circumstances.

PAPER TAPE PUNCH

If the paper tape punch is out of paper, the Monitor outputs the message

!! PPndd MANUAL

on the OC device. The operator corrects the condition and depresses the START key.

If the paper tape punch is off-line or the power is off, the Monitor outputs the message

!! PPndd UNRECOG

on the OC device. The operator corrects the condition and responds to the Monitor with the key-in

PPndd C or R

MAGNETIC TAPE

If an error occurs during the reading or writing of magnetic tape, the Monitor I/O handler attempts a recovery x times (x = NRT, a DCB variable). If the error is irrecoverable, the user is informed via an error return.

If a magnetic tape is addressed and there is no physical reel or power, the Monitor will output the message

!! MTndd UNRECOG

on the OC device. The operator's response depends on the circumstances.

ERROR AND STATUS MESSAGES

When events take place in the system requiring operator intervention, or when one job completes and another job begins, RBM informs the operator of these conditions by messages output to the operator's console (OC device). All such messages from the Monitor begin with two exclamation marks (!!). Generally, these messages require no operator response on the typewriter, but may indicate that some peripheral needs attention.

MONITOR MESSAGES

The messages itemized in Table 4 are output by the Monitor on the OC device.

Table 4. Monitor Messages

Message	Meaning
!!KEY ERROR	Monitor cannot recognize an unsolicited key-in response. A new key-in should be attempted.
!!JOB ABORTED at yyyyy	Background job has been aborted. The "yyyyy" parameter contains the hexadecimal address of the last instruction executed in the background. If aborted because the specified limit on a !LIMIT control command has been reached, yyyyy will contain the word "LIMIT".
!!PAUSE comments	A !PAUSE control command card has been read. The comments field may contain tape mounting instructions. A key-in of "C" after pressing the INTERRUPT switch will cause RBM to continue reading from the job stack.
!!BEGIN WAIT	Background has executed a "WAIT" request; an unsolicited key-in of "C" will continue background processing.
!!BCKG CKP	Background has been checkpointed as a result of a foreground program load.
!!BKG RESTART	Background has been restarted from its point of interruption.
!!yyndd WRT PROT	Indicated unit is write-protected. If a magnetic tape, insert the write ring and place tape in "start" condition to continue if the program is waiting for operator action; or abort background program if the background is performing an invalid operation.
!!yyndd UNRECOG	Some condition on device type yy with physical device number ndd (hexadecimal) has caused device to become not operational.

Table 4. Monitor Messages (cont.)

Message	Meaning
!!yyndd ERROR	A parity or transmission error has occurred on this device. Any automatic retries that were specified have been performed before this message was output.
!!yyndd MANUAL	Device specified is in manual mode and may be out of paper, cards, or tape.
!!RLS NAME NA	A key-in request has been made to release a foreground program, but the name of the program is not included among the active foreground programs.
!!FILE NAME ERR	A problem has occurred from a STDLB key-in request in attempting to open or close a RAD file.
!!FGD AREA ACTIVE	An FMEM key-in request cannot be honored because a foreground program is still active in the area being released.
!!NOT ENUF BCKG SPACE	Insufficient background space to load the requested background program.
!!UNABLE TO DO ASSIGN	An !ASSIGN command cannot be fulfilled because either the DCB cannot be found or the DCB is only five words in length, and a seven-word DCB is required (seven-word DCBs are required for any RAD file assigns).
!!BKG IN USE BY FGD	Background space is being used by the foreground, but a checkpoint was not required since the background was inactive at the time of the foreground load.
!!CK AREA TOO SMALL	An attempt was made to checkpoint the background, but not enough space was available on the CK area of the RAD. The background space will nevertheless be released to the foreground, and the active background job will be aborted when the background is restarted.
!!I/O ERROR ON CKPT	An attempt was made to checkpoint the background, but a RAD I/O error occurred during the process. The background space will nevertheless be released to the foreground, and the active background job will be aborted when the background is restarted.
!!LOADED PROG xxxxxxxx	The specified foreground programs have been loaded for execution by the foreground loader. A maximum of three program names will be output in the one message.
!!UNABLE TO CLOSE DCB xxxxxxxx	The specified DCB was not closed upon releasing a foreground program.
!!PROG xxxxxxxx RELEASED	The specified foreground program has been released.
!!FGT FULL, CAN'T LOAD xxxxxxxx	The specified foreground program cannot be loaded for execution because no room exists in the Foreground Programs Table.
!!CORE USED, CAN'T LOAD xxxxxxxx	The specified foreground program cannot be loaded for execution because the core space required for its execution is already in use.
!!I/O ERR, CAN'T LOAD xxxxxxxx	An I/O error occurred in attempting to load the specified foreground program for execution.
!!NONEXIST., CAN'T LOAD xxxxxxxx	The specified foreground program cannot be loaded for execution because it does not exist on the RAD or a Public Library required by the program does not exist on RAD. The foreground program must exist in the FP area or the OV file.
!!PUB LIB, CAN'T LOAD	The request to load the specified Public Library for execution is not valid, since all Public Libraries must be automatically loaded by the system, as needed.
!!UNABLE TO LOAD BCKG PUB LIB	The current attempt to execute a background program has failed, because the Public Libraries required by the background program could not be loaded. The current background job is aborted.

Table 4. Monitor Messages (cont.)

Message	Meaning
!!CKPT WAITING FOR BCKG I/O RUNDOWN	The checkpoint function is waiting for all background I/O to run down so that the checkpoint of background can be completed.
!!UNABLE TO TRIGGER CONTROL TASK INT.	This alarm is output to OC after the system is booted from the RAD if the RBM Control Task Interrupt cannot be triggered.
SIGMA 5/7 RBM-2, VERSION xxxx	This message is output on the OC device every time the system is booted from the RAD. The message can be terminated prematurely by hitting the BREAK key on the typewriter.

TRAP HANDLER MESSAGES

The following messages are output by the trap handler upon occurrence of the various traps if the user does not specify his own trap handling:

!!MEM. PROT. ERR AT xxxxx

!!PRIVILEGE INST. AT xxxxx

!!NONEXIST. ADD. AT xxxxx

!!NONEXIST. INST. AT xxxxx

!!UNIMPLE. INST. AT xxxxx

!!STACK OVERFLOW AT xxxxx

!!ARITH. FAULT AT xxxxx

!!WDOG TIMER RUNOUT AT xxxxx

!!ILL. PARAM., CAL AT xxxxx

Note that the message "ARITH. FAULT AT xxxxx" is output for the fixed point arithmetic overflow trap, the floating-point fault trap, and the decimal arithmetic fault trap. The message

"!!ILL. PARAM., CAL AT xxxxx"

is output if a user program furnishes the Monitor an invalid parameter while attempting to use a Monitor function.

JCP MESSAGES

In general, the messages itemized in Table 5 are output by the Job Control Processor on both the OC and LL devices. The JCP reads and processes each control command until it encounters a request to execute a processor or user program at which time the appropriate program is read into the background and given control. JCP status or error messages deal with control commands or other program input diagnostics.

Table 5. JCP Messages

Message	Meaning
!!JCP	The JCP has just begun to read control commands. This occurs both at the beginning of a job and between steps within a job. If C is assigned to the typewriter or if "TY" override is in effect, the input light on the typewriter will indicate that RBM is ready for input of a control command. This message is output only to OC.
!!CC ERROR IN ITEM xx	An error exists in a JCP control command in the indicated item. Every item except the ! character followed by a blank or comma is counted in determining the item in error. If the first character is not an exclamation character, the message will designate ITEM0.
!!SCHING FOR JOB CMD	The present job has been aborted and the JCP is searching the job stack for the next JOB or FIN command.
!!CC ERROR, FG KY-IN REQUIRED	A request has been made to run a foreground program from the background job stack without previously inputting an FG key-in. The RUN or ROV command must be reentered after the FG key-in is input.
!!CC ERROR, BT OVERFLOW	The file size input on an !ALLOBT command is greater than the available Background Temp RAD space.
!!FGT FULL, CAN'T LOAD xxxxxxxx	The indicated foreground program cannot be loaded because insufficient space exists in the Foreground Program Table.

Table 5. JCP Messages (cont.)

Message	Meaning
!!FILE xxxxxxxx NONEXIST.	The indicated RAD file was never allocated via the RAD Editor or was never written into.
!!PUB LIB, CAN'T LOAD xxxxxxxx	The designated program on the RUN command is a Public Library and cannot be executed via a RUN command.
!!CC ERROR, ILL. REALLOCATION OF BT	An improper ALLOBT command was input to change a Background Temp (BT) scratch file that was designated as a "saved" file prior to this job step.
!!BT OVERFLOW	Insufficient Background Temp RAD space to execute the requested background program. The job is aborted.
!!BI CKSM ERR !!BI SEQ ERR	JCP loader encountered a checksum or sequence error on a binary card during the loading process.
!!ERR, CONTROL BYTE=xx	JCP loader is not equipped to process the indicated control byte.
!!TOO MANY DEF/REF's	JCP loader has encountered more than 255 declarations in the object module being loaded.
!!UNSATISFIED REF xxxxxxxx	Indicated REF was not satisfied during the loading process. This alarm occurs only on LL if no map was requested, or on LO if a MAP was requested.
!!NOT ENUF SPACE FOR LOAD	JCP loader is unable to complete the load because of insufficient background space.
!!TOO MANY DCB's	The maximum number of M: and F: DCBs was exceeded during the loading process. Approximately 27 DCBs can be accommodated by the system. The excess DCBs will not be stored in the DCB table or the RAD header file.
!!ILLEGAL BINARY CARD	An EBCDIC card was read by the JCP loader where a binary card was expected.
!!UNSATISFIED REF's DURING LOAD	This message is typed to the operator on OC at the end of a load if any unsatisfied REFs were encountered during the loading process.
!!BEGIN IDLE	Job Control Processor has read a FIN card, which completes a job stack. The background then goes into an idle state. Processing will resume on a new job stack following an unsolicited key-in of C.
!!EOT ON FILE xxxxxxxx	End-of-Tape status was returned from an attempt to read or write the indicated RAD file.
!!ILL. NEG. ORG ITEM	JCP loader has encountered an origin item that it is not equipped to handle (an origin item that moves the load location counter in a negative direction). The load will be aborted.
!!ILL. DEFINE FIELD ITEM	JCP loader has encountered a define field item that it is not equipped to handle (a define field item that crosses a word boundary). The load will be aborted.
!!TOO MANY CONTROL SECT.	JCP loader has encountered more than one nonstandard control section. The load will be aborted.
!!ILL. EXPRESSION	JCP loader has encountered an expression that it is not equipped to evaluate (a mixed resolution expression). The load will be aborted.

3. SYSTEM CONTROL COMMANDS

The Monitor is controlled and directed by means of control commands. These commands effect the construction and execution of background programs and foreground programs loaded in from the background job stack, and provide communication between a program and its environment. The environment includes the Monitor and the Macro-Symbol, Symbol, FORTRAN IV-H, SL-1, Overlay Loader and RAD Editor processors, the operator, and the peripheral equipment. The two service processors (the Overlay Loader and RAD Editor) have their own subcommands that are defined in the appropriate chapters.

JOB CONTROL PROCESSOR

The Job Control Processor (JCP) reads all Monitor control commands from the device designated by the "C" operational label. The JCP is a special processor loaded into the background by RBM upon the initial "C" key-in. The JCP is also reloaded into the background following each job step within a job. A job step is defined as all control commands required for the setup and execution of a single processor or user program within a job stack. (See Figure 1 for an example of a two-step job.)

The JCP processes each control command until a request is made to execute a processor or user program, at which time the appropriate program is read into the background and given control.

MONITOR CONTROL COMMANDS

CONTROL COMMAND FORMAT

Control commands have the general form

```
! mnemonic specification
```

where

!	is the first character of the record and identifies the beginning of a control message.
mnemonic	is the mnemonic code name of a control function or the name of a processor. The name may begin any number of spaces after the ! character, except an !EOD command.
specification	is a listing of required or optional specifications. This may include key words, labels, and numeric values appropriate to the specific command.

In this manual, the options that may be included in the specification field of a given type of control command are shown enclosed in brackets (actually no brackets are used in control commands) and parentheses are required to indicate the grouping of subfields. For example, see the options given for the LOAD control command.

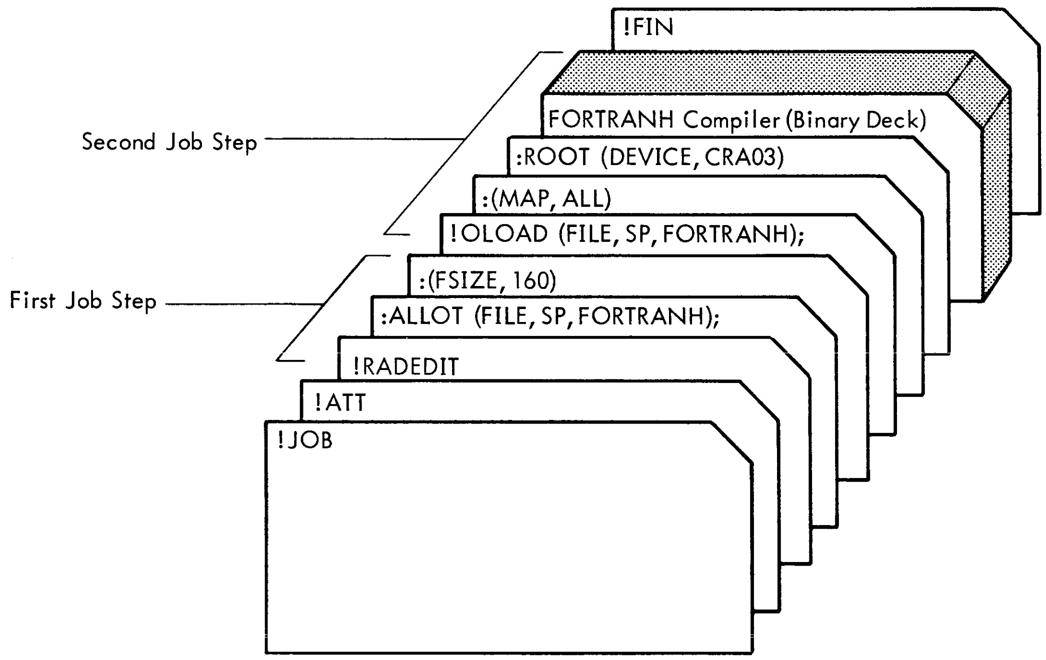


Figure 1. Job Stack with Two Job Steps

SEPARATORS AND TERMINATORS

One or more blanks may separate the mnemonic and specification fields, but no blanks can be embedded within a field. A control command is terminated by the first blank after the specification field; or, if the specification field is absent and a comment follows the command, the command is terminated by a period after a blank that follows the mnemonic field. Annotational comments detailing the specific purpose of a command may be written following the command terminator, but no control command record can contain more than 80 characters.

CONTINUATION CARDS

A control command can be continued from one record to the next by using a semicolon to replace the comma as a subfield terminator in the command's specification field. Column 1 of the continuation cards must contain either an exclamation mark (for a command read by the Job Control Processor) or a colon (for subcommands read by the Overlay Loader or RAD Editor).

If there is an error in a continuation card, the entire command must be re-input after correction.

COMMAND MNEMONICS

For all control commands, the first three characters of the mnemonic following the exclamation character are sufficient to define any mnemonic code or keyword.

Example:

The mnemonics

```
! ALL
! ALLO
! ALLOB
! ALLOBT
```

are all legal representations for the ! ALLOBT control command.

LOGGING CONTROL COMMANDS

Control commands are usually input to the Monitor via punched cards; however, any input device(s) may be designated for this function (see ASSIGN command). All control commands are listed on the output device designated as the listing log (normally a line printer) as they are read. In this manner, the Monitor keeps the operator informed regarding the progress of a job. Control commands that are skipped over until the next JOB command is encountered are listed with a "greater than" character (>) in column 1.

CONTROL COMMAND REPERTORY

All acceptable RBM control commands are given in Table 6, and are listed in a logical, but not necessarily typical, operating sequence. Sample parameters are given in the "Example" column only to illustrate typical parameter formats.

A complete explanation of the parameters is not given. (A more detailed description is given in the XDS Sigma 5/7 RBM Reference Manual.) Note that this table lists only the standard commands recognized by the Monitor; the commands for the two service processors (Overlay Loader and RAD Editor) are given in their respective sections of this manual.

Table 6. RBM Control Commands

General Form	Example	Meaning
! JOB [account number,name]	! JOB 13962,SAMPL	Starts a new job and signals end of any previous job. Resets all operational labels to their standard assignments.
! ATTEND	! ATTEND	Indicates that recovery action is to be attempted at the console in case of program error or abort in the background. An unsolicited C key-in causes the background to continue processing from point of error. If an ATTEND command is not used and a job aborts or error occurs, all commands, binary records, and data are skipped until a new JOB or FIN command is encountered.
! ASSIGN (dcb [area,name]) [,(option),(option),. . . ,(option)]	! ASSIG (M:LO,9TA81),VFC or ! ASSI (M:SI,D1,PRESTORE)	Specifies the physical devices or RAD files to be used to process the current job. Each ASSIGN assigns a Data Control Block name to an operational label (logical device name), a RAD file, or physical device. ASSIGN commands must appear prior to the appropriate RUN or Processor name command and affect only that one job step.
! PMD [U] [,(from,to)] [,(from,to)] . . .	! PMD U,(1200,1300), (2000,3000)	Dumps specified areas of memory on the DO device if a background job is aborted or terminated normally during execution. The U parameter

Table 6. RBM Control Commands (cont.)

General Form	Example	Meaning
		specifies an unconditional dump at the end of the job even if there were no errors. If the (from,to) locations are not specified, the entire background area is dumped. The command must precede the RUN command.
! LIMIT n	! LIMIT 3	Sets the maximum allowable execution time for a background program in minutes. If the job exceeds the limit, it is aborted with a postmortem dump if so specified via a PMD command.
! ALLOBT(FILE,nn)[,option) (,option) . . .]	! ALLO (FILE,X1),(FORMAT, C),(FSIZE,1000),SAVE	Defines the files in the BT (Background Temp) area of the RAD and overrides any JCP default definitions.
! STDLB (label [,area] ,name) [,(label [,area] ,name). . .]	! STDLB(BO,GO),(CO,D2, COMPRESS),(LO,9TA80)	Changes the assignment of an operational label (except operator's console) to a temporary assignment that stays in effect until the next JOB command is encountered.
! LOAD [(option),(option)]	! LOAD(IN,CRA03),(OUT,SP, OLOAD),(SEG,5),MAP	Loads a program on the RAD with absolute linkage for its core execution location. Foreground programs can only be loaded on the OV file or FP area of the RAD.
!EOD	!EOD	Defines the end of each block of data within a data deck. Monitor returns an EOD status when each EOD command is encountered. Any number of EOD commands can be used in a job and for any purpose. There must be no space between the exclamation character and the mnemonic.
! CC	! CC	Removes typewriter override of the C device (see TY key-in description in Chapter 2). The next control command will be read from the C device instead of the typewriter.
! MESSAGE message	! MESSAGE SEND ALL SAVE TAPES TO DICK WEAVER	Types a message to the operator on OC and LL devices. Processing continues without pause after message is output, and no operator intervention at the console is necessary.
! PAUSE	! PAUSE KEYIN SY	Causes the Monitor to enter a WAIT state after a message is output on OC, giving the operator time to carry out the instruction in the message. Processing continues after a (unsolicited) C key-in.
! POOL n	! POOL 3	Overrides the default allocation of blocking buffers for the background. Ccuses n blocking buffers to be allocated.
! RUN area,file name	! RUN BP,SORT	Causes the named program to be executed. The area parameter must be either SP, FP, or BP. The loading of a program into the foreground area via a RUN control command must be preceded by an FG key-in. Remains in effect for a single job step only.

Table 6. RBM Control Commands (cont.)

General Form	Example	Meaning
! ROV	! ROV	The ROV (RUN OV) command causes execution of the program on the OV file. The loading of any program into the foreground area via an ROV control command must be preceded by an FG key-in. A foreground program loaded by !ROV is given the name OV.
! PFIL [area,]name[,BACK] ! PREC [area,]name[,BACK][,n]	!PFIL GO ! PREC D1,ABCD,30	File and record positioning commands used to position a device within its current file. The PFIL command is only valid for magnetic tapes or RAD files and leaves the device positioned before the file mark in the appropriate direction. Only background devices (not dedicated to foreground or IOEX) can be positioned.
! SFIL name[,BACK][,n]	!SFIL 9TA82,BACK,4	Skips one or more files on a magnetic tape unit. It cannot be used to position a RAD file. The command positions the deivce immediately following the specified tape mark in the appropriate direction. Only undedicated devices can be positioned. The n parameter specifies the number of files to skip.
! REWIND [area,]name	! REWIND 7TAEO	Rewinds a magnetic tape or RAD file. It has no effect on other devices.
! UNLOAD [area,][name]	! UNLOAD D4,OUTPUT	Causes the specified magnetic tape to be rewound in manual mode. Operator intervention is required to use the device again (i.e., depressing the ATTENTION and START switches on a tape device). For a RAD file, UNLOAD produces the same results as a REWIND command.
! WEOF [area,] name [,n]	! WEOF 9TA81,2	Causes an end-of-file mark to be written if appropriate to the device. For magnetic tape, a tape mark is written; for a RAD file, a logical file mark is written; for paper tape a !EOD record is written. The WEOF command is ignored for all other devices. The n parameter specifies the number of end-of-file marks to write.
! DAL [PAL]	! DAL PAL	Causes the contents of the accounting log (on the AL file on the D1RAD area) to be dumped. The PAL option causes the AL file to be purged after dumping.
! FIN	! FIN	Specifies the end of a stack of jobs and no jobs are pending. When the FIN command is encountered, it is written out on LO to inform the operator that all current jobs have been completed. The Monitor types out "BEGIN IDLE" on OC and enters the idle state.
! MODIFY (module,loc),value[R]		Special control command used only for modifying (patching) system modules at system boot time. See Appendix E, "System Patching" in the Sigma 5/7 RBM Reference Manual (Publication No. 90 15 81) for complete description and use.

PROCESSORS/MONITOR INTERFACE

All processors, whether service, system, or user, reside on the Systems Program area of the RAD, operate in the background core space, and are called from the Systems Program area for execution by the control command

```
! name parameters
```

where

name is the RAD file name of the processor to be executed. The standard name format for service and system processors is as follows:

Name Format	Processor Called
!OLOAD	Overlay Loader
!RADEDIT	RAD Editor
!MACRSYM	Macro-Symbol
!SYMBOL	Symbol
!FORTRANH	FORTRAN IV-H[t]
!SL1	SL-1

[t]Real-Time FORTRAN IV-H is called by using the RT option in the parameters of a !FORTRANH command.

parameters are optional parameters interpreted by each processor except the !RADEDIT command, which does not need parameters. The options for all system processors recognized by RBM are defined in Table 7. (See "Related Publications" at the beginning of this manual for manuals giving a more detailed description of the system processors.)

A typical example of a processor called in by a user's source program for an assembly is given below.

Example:

```
! MACRSYM SI,LO,CI,BO
```

This control command specifies that control is to be given to the Macro-Symbol assembler. The SI parameter specifies that symbolic (source) input is to be read from the device to which the SI operational label is assigned; the LO parameter specifies the device to which the listed output is to be written; the CI operational label specifies the device from which the compressed input is to be received; and the BO operational label specifies the device to which the binary output is to be transmitted.

Table 7. Processor Specifications Options

Specification	Use	Used By
BA	Selects batch assembly mode	Macro-Symbol
BO	Relocatable binary output (on cards or paper tape) on the BO device.	FORTRAN IV-H, SL-1, Symbol, Macro-Symbol
CI	Compressed input from the CI device	Macro-Symbol
CN	Concordance listing	Symbol
CO	Compressed output on the CO device	Macro-Symbol
D	Debug mode compilation	FORTRAN IV-H
GO	Relocatable binary output to temporary RAD storage (i.e., the GO file)	Symbol, Macro-Symbol, SL-1
LO	Listing output produced on the LO device	FORTRAN IV-H, Symbol, Macro-Symbol, SL-1
LS	Source Listing produced on the LS device	FORTRAN IV-H, SL-1
LU	Listing of the update decks (if any) produced on the LO device	Macro-Symbol
S	S in column 1	FORTRAN IV-H, SL-1
SI	Symbolic input from the SI device	Macro-Symbol, SL-1
SO	Symbolic (source) output produced on the SO device	SL-1

4. RUNNING BACKGROUND JOBS

JOB DEFINITION

The fundamental unit in background processing is the job. A typical job might consist of the following elements:

1. A JOB command card to signal the beginning of a new job. The Monitor resets all operational labels to their permanent assignments.

2. Optional ASSIGN or STDLB commands that define to the Monitor the peripheral devices and RAD files that are to be used for I/O. These commands are only needed to change the inherent I/O assignments in a program.

3. A processor command that calls in the correct system processor (e.g., Macro-Symbol assembler), user program (or combination), or a RUN command to execute a program.

4. Any source data the program is designed to process. The data may be contained in a card deck, magnetic or paper tape, or on a RAD data file.

A number of optional control commands can be included in a job, such as a LIMIT command to set the allowable execution time for a given program. Figure 2 illustrates the simplest case of a deck setup for a job.

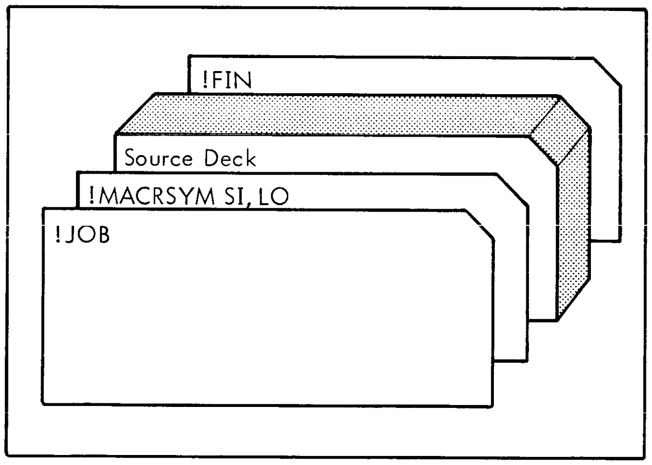


Figure 2. Source Program Assembly Example

In Figure 2, the symbolic input is received from the SI device and the listing output is produced on the LO device.

LOAD AND GO JOBS

"Load and Go" jobs are programs that immediately go into execution mode when the source program is successfully assembled or compiled. That is, the object program is loaded into core from a temporary file on the RAD when the assembly is completed instead of being manually loaded from a card or tape device by the operator. Figure 3 illustrates a typical "Load and Go" job.

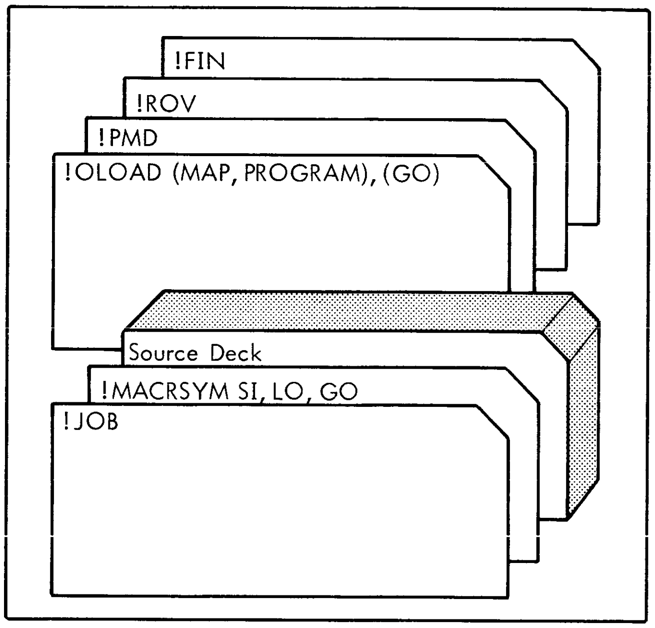


Figure 3. Load and Go Deck Example

In Figure 3, the binary object program produced from the assembly is placed in a temporary (GO) file from which it is later loaded into core for execution. The resultant file is always temporary and can not be retained from one job to another. The Overlay Loader will load the program root into the OV file for execution. A postmortem dump is specified.

BATCH JOBS

Batching permits a user to load a series of source or compressed programs for assembly or compilation under a single system processor control command. The parameters specified on the processor command will hold true for every job within the batch.

When batch assemblies consist of successive updates from card input to compressed programs from the RAD or tape, each update packet is terminated by a +END card, not by an !EOD card. There must be a one-to-one correspondence of update packets to compressed programs. If there are no updates to a particular program in a batch, the "missing" update packet must be represented by a +END card inserted in the proper place in the update deck. The example illustrated in Figure 4 is a typical batch job.

In Figure 4, successive assemblies are performed with a single MACRSYM command until a double EOD is read. The device assignments and options on the MACRSYM command apply to all assemblies within the batch. A program assembly is considered terminated when an END Macro-Symbol directive is processed.

JOB STEPS

A job step is the execution of a program (system processor or user program) in the background. The processors that operate

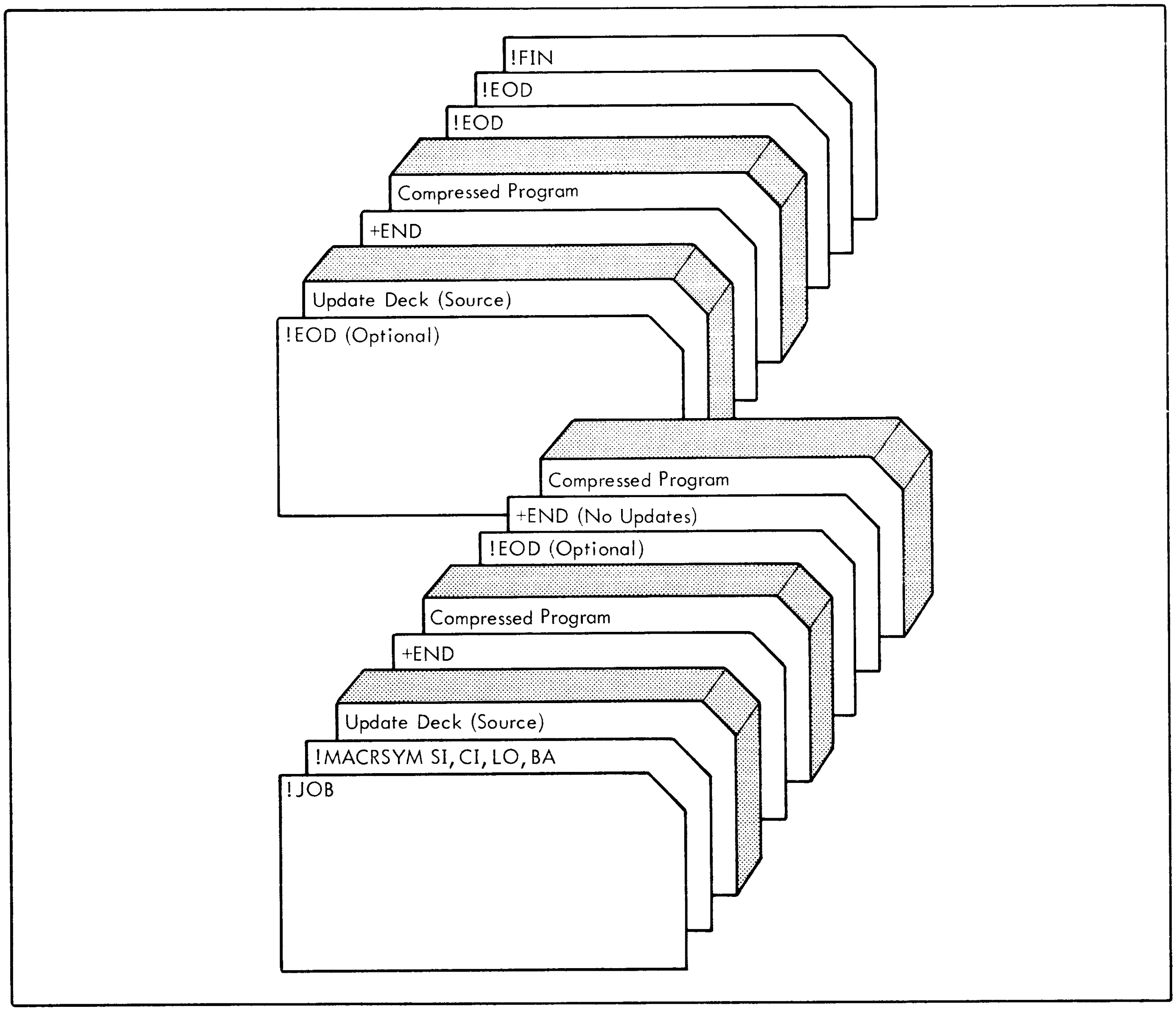


Figure 4. Batch Job Example

under RBM are service processors (RAD Editor and Overlay Loader) language processors, and user processors. Service and language processors are supplied with the RBM system. User processors are created at the local installation.

GENERAL OPERATING CONSIDERATIONS

The basic types of background jobs the operator may encounter fall in the following general categories:

1. Assemblies or compilations of original source programs (written in some symbolic language such as FORTRAN IV-H or Macro-Symbol) into object modules. The output from such a job might include a source listing, an object listing, and an object program deck.

2. Trial executions of partly debugged programs called "test cases". Sample data is loaded and the program processes the data under various conditions. Additional control command options may be used to vary the processing of data or format of the final output.

3. Processing data by an operational program. Such programs generally reside in the Background Programs area of the RAD if they are frequently or regularly used. Less frequently used programs may be loaded from a card reader or tape device. Data to be processed can be located in the foreground or background data areas of the RAD or input from some other external source.

BACKGROUND JOB RESTRICTIONS

The Monitor imposes two fundamental restrictions in processing background programs:

1. Background programs are given CPU time only after real-time hardware interrupts are satisfied. That is, nothing can take place in the background that inhibits interrupts or in any way interferes with real-time responsiveness. Thus, background programs will not be guaranteed any processing time if the foreground is very active.

2. The Monitor prevents any attempt by a background program to write into or execute instructions in foreground core storage, to write into foreground RAD file areas, or to utilize devices or services dedicated to real-time tasks. Any attempt by the background program to violate this protection or to execute privileged instructions, either intentionally or through program error, may result in the Monitor aborting the background program.

There is no read protection for the foreground areas, and background programs can read from real-time core storage and secondary (RAD file) storage without restriction. A typical example of using the read feature would be a real-time data acquisition program that accumulates real-time data and writes it out on the foreground data file area of the RAD. A background program could then be loaded at a later, less critical period to read and further process the data without disturbing the raw data that might be needed for some further real-time task.

Frequently used user programs reside on the RAD in load module form in the BP area and are called via a !name command, where name is the name of the file.

ERRORS IN JOB STREAM

In running assemblies or compilations, those errors from which the operator can recover are generally control command sequence or format errors (i.e., a mispunched card); when processing data with an operational program, data format errors are a frequent cause of job abort and it may be necessary to take a dump of the data file.

During processing of the job stream, the Monitor will go into a WAIT state after outputting a request on the operator's console to ready some device or load data input.

OPERATOR INTERRUPT DURING CONCURRENT FOREGROUND BACKGROUND OPERATIONS

If the operator depresses the INTERRUPT switch during concurrent foreground/background operations (perhaps to change the status of a background job), the Monitor will not acknowledge the interrupt with a !!KEY-IN message until all current and waiting foreground processes are completed. This is because the Control Task that processes the key-in is connected to the lowest priority interrupt in the system. For the same reason, the operator cannot alter the current foreground task.

ATTENDED RUNS

Attended runs are frequently specified when new programs are submitted for processing. An !ATTEND control command in the user's job will inhibit the Monitor ABORT routine, and the Monitor will go into a WAIT state if it encounters an error it cannot correct. This gives the operator time to take memory dumps or initiate recovery action, if possible. If recovery procedures cannot be successfully completed, the operator's only recourse is to abort the job through an "X" key-in after taking any specified dumps.

5. RUNNING FOREGROUND JOBS

OVERVIEW

Operator control and manipulation of foreground programs requires a knowledge of some of the characteristics of such programs and of how core memory is partitioned. The positioning of memory into various areas such as the systems area, foreground area, foreground blocking buffering area, background area, background buffering area, etc., is installation-determined during SYSGEN and becomes the standard default allocation. In a typical installation, core memory will be allocated as is illustrated in Figure 5.

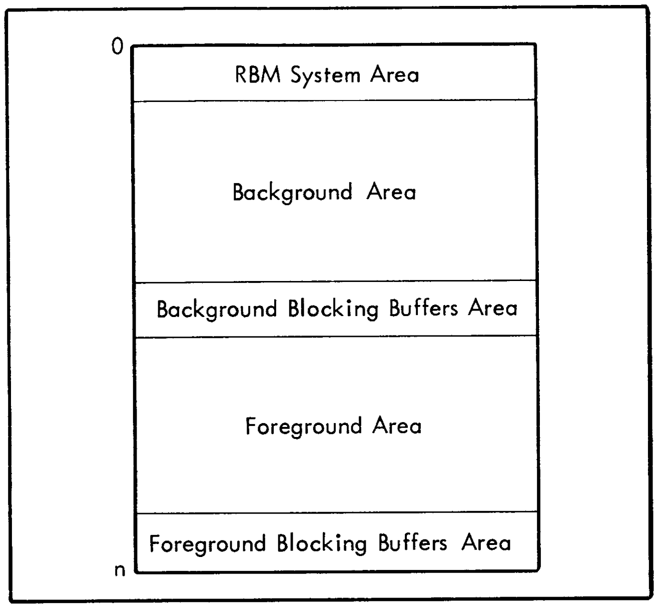

0
RBM System Area
Background Area
Background Blocking Buffers Area
Foreground Area
Foreground Blocking Buffers Area
n

Figure 5. Typical Core Memory Partitioning

The boundary of the background blocking buffers area is not static, but can shift into the background program area, depending upon the number of blocking buffers present, which can vary from job step to job step. The first-word address (FWA) of the foreground memory area can be moved up into the background through an FMEM key-in if more foreground core space is required.

Several foreground jobs can be resident simultaneously in protected core; the number that can be resident is determined by the number of entries in the Foreground Program Table established during SYSGEN.

REAL-TIME INTERRUPTS

Foreground programs are frequently loaded and initialized at System Boot-time. Program execution is initiated when a signal is received from some external source, such as telemetry equipment, factory equipment, or a medical device in a hospital. The signal causes triggering of an interrupt that initiates execution of its connected task (a body of code within the program).

Each foreground program in core will be connected to one or more interrupts. While these interrupts are generally triggered from outside the system, the signal can be internal; that is, one foreground program can trigger execution of another program or the signal can be received from an internal interval timer (real-time clock). Each interrupt has a different priority level, and when more than one interrupt is in a waiting state, this priority level determines which program will next become active.

TEMPORARILY INACTIVE FOREGROUND

A foreground program may be resident in core with all interrupts armed and enabled, but be temporarily idle. That is, it can be waiting for a new interrupt to be triggered before resuming execution. Care must be exercised in reallocating core when such programs are present in the system, since the apparent lack of activity may lead to a belief that they have completed. When the program is released, the message

```
!!PROG xxxxxxxx RELEASED
```

will output on the OC device. Space will now be available for a different foreground program, or it may be possible to move the foreground boundary to make more core space available to the background.

CONSOLE INTERRUPT PRIORITY LEVEL

To prevent accidental interference with critical real-time processes, the Console Interrupt Task triggers the RBM Control which is normally the lowest priority interrupt in the system. This prevents the Monitor from processing a key-in until the active interrupt and any pending interrupts have been processed. The system will then acknowledge an operator initiated interrupt with a

```
!!KEY-IN
```

message on the OC device.

RUNNING FOREGROUND PROGRAMS

Before any foreground program can be loaded into core for execution, it must have been previously loaded onto the RAD in absolute core image format, where it will reside either in a file in the FP (Foreground Programs) area or on the OV file. The request to RUN the program is initiated in one of four ways:

RUN name key-in, where name is the name of a program in an FP area

ROV key-in, to run the program currently on the OV file

!RUN name control command, read in from the background job stack on the CC device for the named program in the FP area. The command must be preceded by an FG key-in.

!ROV control command, read in from the background job stack for a job located in the OV file (Only one job at a time is permitted on the OV file and is always called OV.) The command must be preceded by an FG key-in.

When the RBM system encounters the request to RUN, it performs the following steps:

1. LOAD: The RBM system reads the program header, which specifies the program's required core location and size, and tests if the required core space is available. If the foreground program extends into the current background core space and a background program is active, the background program is checkpointed and the foreground program is read into core from the RAD. The message

   !!LOADED PROGRAM xxxxxxxx

   will be output on OC.

2. INITIALIZATION: Control is transferred to the program entry at the RBM Control Task priority level. The steps the program then follows depends on its structure. Typical steps for execution could be the following:

   a. Arm and enable interrupts.

   b. Set up interval timers (the timers define the time delay between the triggering of interrupts).

   c. Connect the tasks to their interrupts.

   d. Trigger the interrupts.

RELEASING FOREGROUND PROGRAMS

Release is accomplished through either operator key-in (RLS) if he wants to bump the program for a different program that requires some of the same core space, or through a system call from the active foreground program. In any event, the following actions will take place:

a. All interrupts used in the program are disarmed, disabled, and disconnected.

b. The memory space used by the program (both foreground and any background) is marked as unused.

c. The checkpointed background program (see below) is restarted unless the core space is needed by a new foreground program.

To release an active foreground program and load another foreground program through the RLS key-in, proceed as follows:

1. Depress console INTERRUPT key. This causes a flag to be set and the RBM Control Task to be triggered. Since the Control Task Interrupt is the lowest priority in the system, it will not be recognized until the active foreground interrupt and any waiting interrupts have completed their execution cycle. The system will then output the

   !KEY-IN

   message on the OC device.

2. Key-In

   $\text{RLS}\begin{Bmatrix}\text{OV}\\ \text{name}\end{Bmatrix}$

where

OV is the foreground program currently in the OV file.

name is the name of a currently active program that is located in the FP area of the RAD.

When the active program is successfully released, the system will output the message

!!PROG xxxxxxxx RELEASED

on the OC device.

CHECKPOINTING BACKGROUND

When real-time programs are running concurrently with background programs, a foreground program can borrow core storage space from the active background program if necessary. When a request is made to RUN a foreground program that requires some portion of memory currently in the Background area, the system will borrow the background space for the foreground by storing the background area on secondary (RAD) storage and saving the status. This procedure is called checkpointing. The system will output the

!!BCKG CKPT

message on the OC device to inform the operator that checkpoint has taken place. No operator action is required.

If background I/O is taking place when the checkpoint is attempted, the system will immediately output the message

!!CKPT WAITING FOR BCKG I/O RUNDOWN

on OC, since the foreground cannot seize background memory until the current background I/O has completed. When the I/O has terminated, the checkpoint proceeds.

RESTART

When the foreground program is released and no longer requires the borrowed background memory space, the system will restart the background job step by reading the saved image from secondary storage and resuming execution. It will then output the message

```
!!BCKG RESTART
```

on OC to inform the operator that the checkpointed program has been restarted from its interrupted point. No operator action is required.

PREVENTING CHECKPOINT

When a foreground program is loaded into core for execution, it is sometimes useful to prevent checkpoint of a background program. This can be done by allocating sufficient space to the foreground through the FMEM key-in. The operator keys-in

```
FMEM y
```

where y is the new number of pages to be allocated to the foreground. Any change in memory allocation as a result of the FMEM key-in does not actually take place until the conclusion of the current background job step.

RESTORING MEMORY

When the foreground program is released, either through normal completion or operator RLS key-in, the operator may restore the background/foreground boundary through an

```
FMEM y
```

key-in, where y is the number of pages of memory to be allocated to the foreground.

Should any foreground program be resident in any portion of memory being returned to the background, the message

```
!!FDG AREA ACTIVE
```

will be output by the system, and memory will not be returned until the foreground program is released.

OPERATOR INTERVENTION

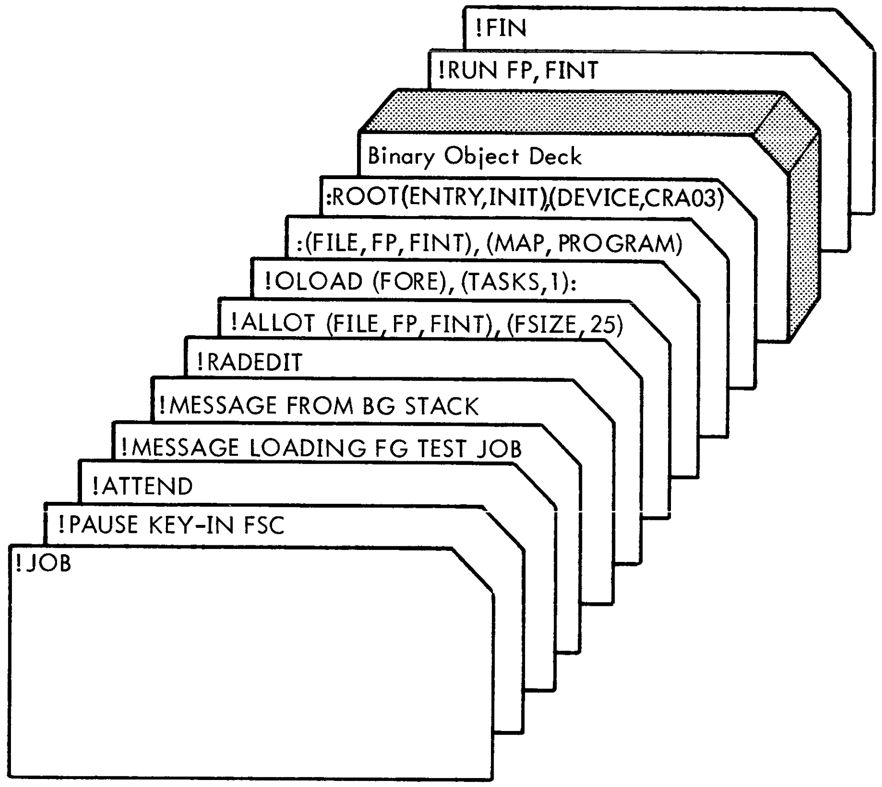
Foreground programs already operational in the system almost never require operator manipulation. Occasionally, the Monitor may output a status message that the background has been checkpointed or restarted, or request the operator to ready some special purpose device.

ERROR RECOVERY

Error recovery for operational foreground programs is impossible in most cases, and the operator's only recourse is to call the designated person or Customer Engineer (for a clearly evident machine malfunction).

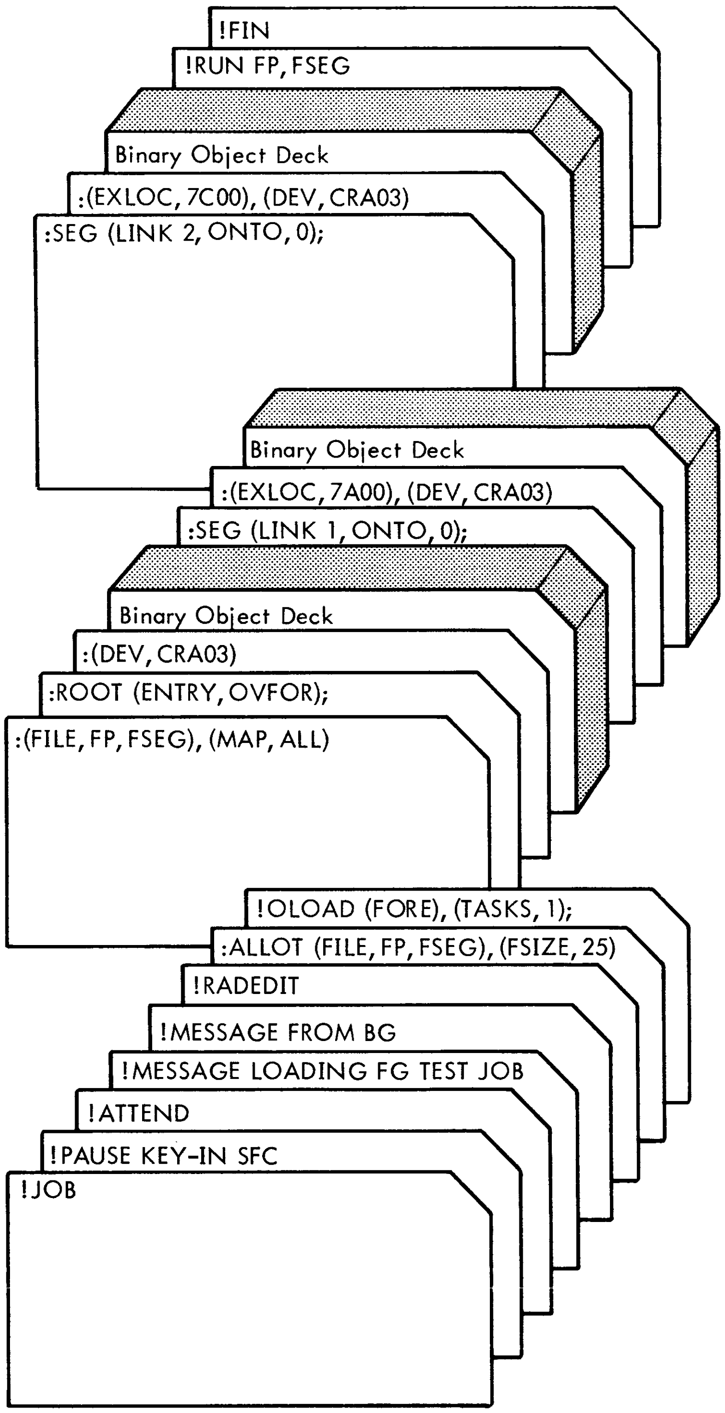
LOADING NEW FOREGROUND JOBS

New foreground programs can be loaded into the foreground programs area of the RAD from the background job stack without a new SYSGEN. Operator handling of such programs is identical to background jobs except that loading must be preceded by an SY key-in to access protected RAD areas. Figures 6 and 7 illustrate typical examples of loading foreground programs from the background job stream.


!FIN
!RUN FP, FINT
Binary Object Deck
:ROOT(ENTRY,INIT),(DEVICE,CRA03)
:(FILE, FP, FINT), (MAP, PROGRAM)
!OLOAD (FORE), (TASKS,1):
!ALLOT (FILE, FP, FINT), (FSIZE, 25)
!RADEDIT
!MESSAGE FROM BG STACK
!MESSAGE LOADING FG TEST JOB
!ATTEND
!PAUSE KEY-IN FSC
!JOB

In this example, the RAD Editor allots a file (FINT) in the Foreground Programs (FP) area of the RAD. The Overlay Loader (see Chapter 7) loads the binary object deck in the file FINT in core image format. The !RUN control command causes execution of the foreground program. A PROGRAM map is specified.

Figure 6. Load and Execute Foreground Program



In this example, the RAD Editor allots space for a file called FSEG in the Foreground Programs (FP) area of the RAD. The Overlay Loader (see Chapter 7) loads a root and two segments into FSEG in core image format. Following an operator FGC key-in, the overlaid program is executed via the !RUN control command. An ALL map is requested.

Figure 7. Load And Execute Segmented Foreground Program

6. RAD EDITOR OPERATIONS

Before any job or data can be loaded on the RAD for execution, it must be allocated file space in the appropriate RAD area. The RAD Editor is called in by the user's job to perform this function via a !RADEDIT control command. Subsequent :ALLOT subcommands read by the Editor define the file or files to be used.

The RAD Editor maintains order in all the permanent RAD areas through use of file directories for each area. As files are added to a given area, the Editor generates a new file entry in the appropriate directory so that files can be located on demand. The permanent RAD areas that have file directories are

Foreground Programs area

Background Programs area

System Programs area

Data areas (foreground and background)

In addition to allocating space for new entries, the RAD Editor performs the following functions:

Deletes entries from the permanent file directories.

Enters data files on the foreground or background data areas.

Compacts permanent file directories and RAD areas.

Truncates empty space from the end of files.

Maps permanent RAD file allocations.

Dumps contents of RAD files or areas.

Copies permanent RAD files.

Copies object modules contained in libraries.

Saves contents of RAD areas on a magnetic or paper tape device (SAVE tape) in self-reloadable form.

Restores previously saved RAD areas to their RAD location.

Maintains library files on the RAD for Overlay Loader use.

Zeros out (clears) complete RAD areas.

Temporarily inhibits and restores use of bad RAD tracks in permanent areas.

RAD AREAS PROTECTION

Software protection of the SP, FP, BP, and foreground data areas of the RAD is provided by requiring the operator to key-in "SY" before any of these areas are modified by a background processor. The only areas that can be modified that do not require an SY key-in are the Background Data areas. The message

!! yyndd WRT PROT

or

!! PAUSE KEY-IN SY (if included in the job stack)

will be output on OC to inform the operator that access to a protected RAD area is requested.

CALLING RAD EDITOR

When an ! RADEDIT control command is read from the C device, the RAD Editor is loaded into core memory from the RAD. Control is transferred to the RAD Editor which reads commands that specify the functions to be performed.

COMMAND FORMATS

All RAD Editor commands are input from the C device and listed on LL. The general form for RAD Editor commands is identical to the Monitor control command format described in Chapter 3, with the symbols below being used to aid in describing the RAD Editor commands given in Table 8.

aa refers to a permanent RAD area and must be one of the following:

- BP is the Background Programs area.
- D1 through DF is the Background and Foreground Data areas.
- FP is the Foreground Programs area.
- SP is the System Programs area.
- zz refers to any RAD area

nnnnnnnn refers to a file name or library module (maximum name length of eight alphanumeric characters).

yyndd refers to a physical device name, where

- yy specifies the type of device: CR, CP, etc.
- n specifies the IOP number: A for IOP0, B for IOP1, etc.
- dd specifies the device number: 03, 80, etc.

op refers to an operational label: BI, SI, etc.

All RAD Editor commands are itemized in Table 8. The entries in the "Example" column are given only to illustrate typical command formats. For brevity, detailed explanations of the parameters for each command is not given (a detailed description for each command is given in the XDS Sigma 5/7 RBM Reference Manual).

Table 8. RAD Editor Commands

General Form	Example	Meaning
!RADEDIT	!RADEDIT	Call the RAD Editor into core to read subcommands.
:ALLOT (FILE,aa,nnnnnnnn)[,(option)] [,(option)] [,(option)]. . . [,(option)]	:ALLOT (FILE,BP,TEST,(FORMAT,U), (FSIZE,50),(RSIZE,90)	Add a new entry to specified permanent file directory that allocates space for a new file.
FROM :COPY $\left(\begin{Bmatrix}\text{FILE,aa [,nnnnnnnn]}\\ \text{LIB,aa,nnnnnnnn}\\ \text{IN,}\begin{Bmatrix}\text{op}\\ \text{yyndd}\end{Bmatrix}\end{Bmatrix}\right)$ TO $\left(\begin{Bmatrix}\text{FILE,aa [,nnnnnnnn]}\\ \text{LIB,aa}\\ \text{OUT,}\begin{Bmatrix}\text{op}\\ \text{yyndd}\end{Bmatrix}\end{Bmatrix}\right)$ [, VFC][, ADD] [,BIN][, CC][,FBCD]	:COPY (IN,CRA03),(FILE,D1,XYZ) !COPY (FILE,D1,XYZ),(OUT,9TA80)	Copy single data files or modules from one device to another. If nonstandard binary (BIN) or control commands (CC) are copied from the C device, operator must assign the C device to 0 when the message !!KEY-IN STDLB C,0 is output, and reassign when !!COPY ENDED appears. An !ATTEND command must be present in the job when the BIN and CC options are specified.
:DELETE $\left(\begin{Bmatrix}\text{LIB}\\ \text{FILE}\end{Bmatrix}\text{,aa,nnnnnnnn}\right)$	:DELETE (FILE,BP,TESTA)	Delete a file directory entry and file from specified permanent RAD area, or an object module from designated library.
:CLEAR zz,zz	:CLEAR D1,DF	Zero out (clear) specified RAD areas.
:SQUEEZE aa,aa,aa,... or :SQUEEZE ALL	:SQUEEZE SP	Regain unused space within permanent RAD areas resulting from file deletions and truncations, and library module deletions.
:TRUNCATE (FILE,aa,nnnnnnnn [,g]) ,(FILE,aa,nnnnnnnn [,g])..., (FILE,aa,nnnnnnnn [,g]) or :TRUNCATE aa,aa,aa,...	:TRUNCATE (FILE,BP,TEST)	Truncate empty space from the end of specified file(s) by allocating space equal to the actual length of the file. For a direct access file, the length of the file must be specified in granules g.
:MAP aa,aa,aa,... or :MAP ALL	:MAP BP,D4	Map the specified permanent RAD areas to the LO device.

Table 8. RAD Editor Commands (cont.)

General Form	Example	Meaning
:DUMP (FILE,aa,nnnnnnnn) [,SREC,value)] [,EREC,value)] or :DUMP zz [,(SREC,value)][,(EREC,value)]	:DUMP (FILE,BP,TEST) :DUMP BP,(SREC,6),(EREC,9)	Dump designated random or sequential access file onto the specified LO device in hexadecimal. All permanent RAD areas, plus the IOEX access can be dumped. The first form of the :DUMP command dumps specified files; the second form dumps RAD areas.
:SAVE zz,zz,. . . or :SAVE ALL	:SAVE SP,BP,D2	Dump specified RAD area(s) onto the BO device (must be magnetic or paper tape) for subsequent restoration. If BO is magnetic tape, it is rewound and data saved is verified. If tape verifies correctly, the message, 'SAVE TAPE OKAY' is output. The first form of the :SAVE command can specify any RAD area; the second form includes all RAD areas except Background Temp area and Checkpoint.
:RESTORE zz,zz, . . .	:RESTORE SP,BP,D2	Restore specified, permanent RAD areas saved via a :SAVE command. Read after write is employed to verify restored data.
:BDTRACK yyndd,number[,number]. . .	:BDTRACK DCAF0,10,11	Inhibit RAD Editor's use of specified number of tracks (hexadecimal) on the designated RAD. A track containing a sector of the file directory is not permitted to be removed from use.
:GDTRACK yyndd,number [,number]	:GDTRACK DCAF0,10,11	Restore RAD and specified hexadecimal number(s) of the tracks previously inhibited from use by :BDTRACK command.

RAD EDITOR DECK SETUPS

Typical decks involving the use of the RAD Editor are illustrated in Figures 8 and 9.

In Figure 8, the CO operational label is assigned to a RAD file called COMPRESS in background data area D1 of the RAD. The compressed output is written on the COMPRESS file.

In Figure 9, the RAD Editor allots file XYZ to receive the card input to be copied. (Note that the RSIZE parameter on the :ALLOT command must contain 30 instead of 20 for binary card input.) The :COPY command specifies that input is to be read from the card reader and copied to the XYZ file. The next :COPY command specifies that a copy of the card input is to be output to the card punch. The !REWIND command rewinds the magnetic tape assigned to 9TA80. When the RAD Editor encounters the !REWIND command, it releases control to the Monitor, so the next !RADEDIT command calls the RAD Editor back in to copy file XYZ to the magnetic tape. The Monitor writes an EOF on the tape and rewinds the tape.

RAD EDITOR ERROR MESSAGES

The RAD Editor outputs error messages on the OC and LL devices. If OC and LL are assigned to the same device, duplication of messages on LL is suppressed. If an operator response is required, the RAD Editor will call the Monitor "WAIT" routine. The operator initiates a console

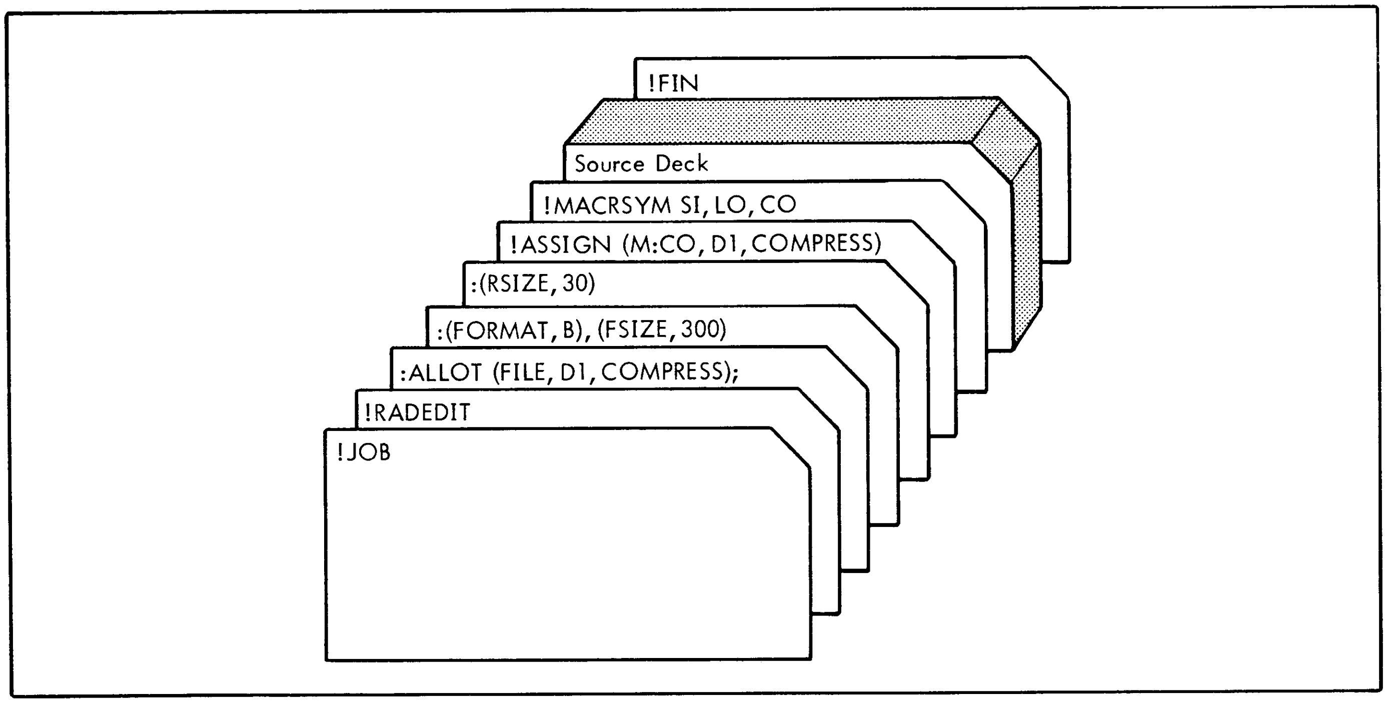

!FIN
Source Deck
!MACRSYM SI, LO, CO
!ASSIGN (M:CO, D1, COMPRESS)
:(RSIZE, 30)
:(FORMAT, B), (FSIZE, 300)
:ALLOT (FILE, D1, COMPRESS);
!RADEDIT
!JOB

Figure 8. RAD Editor ALLOT Example

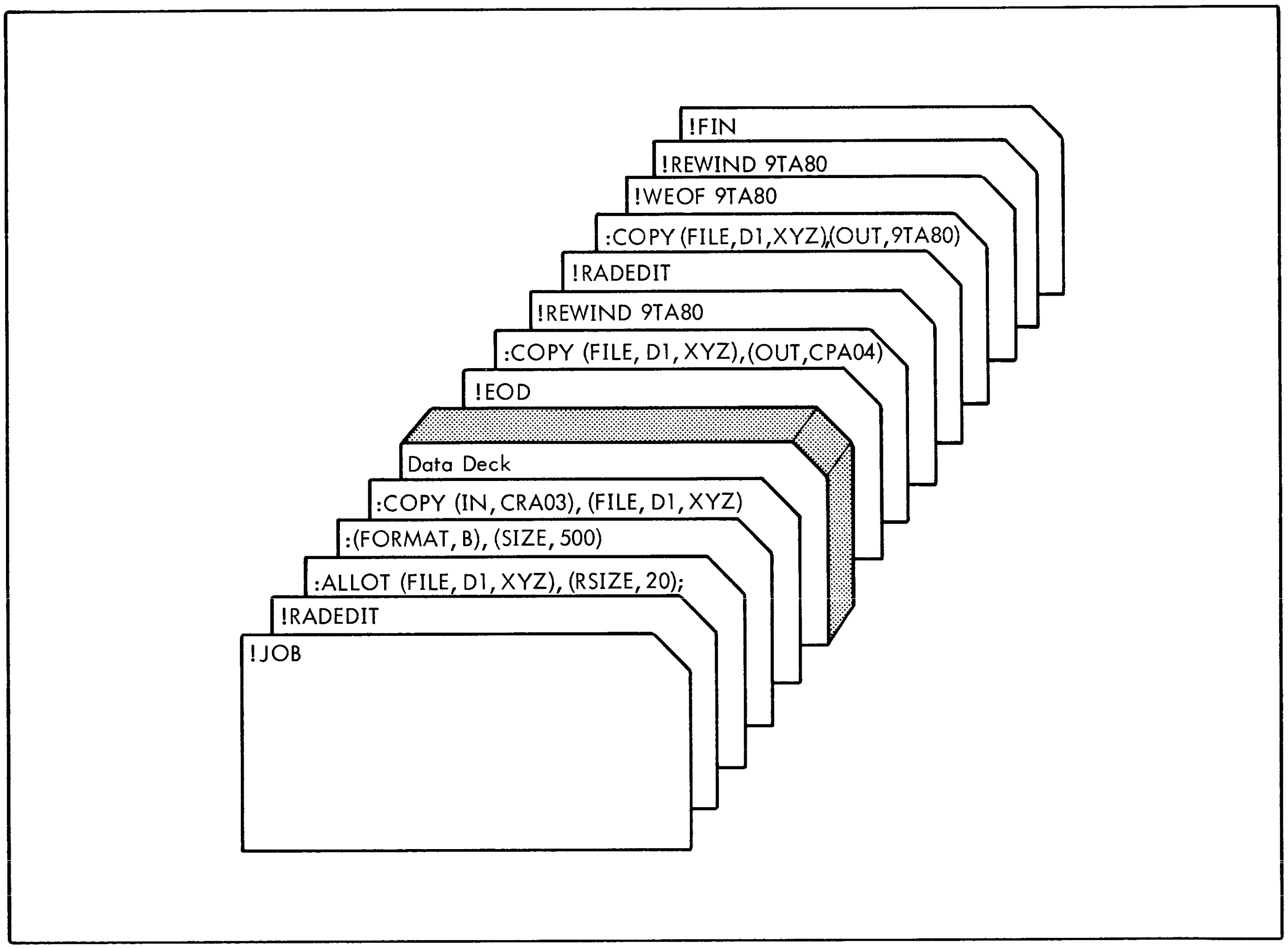

!FIN
!REWIND 9TA80
!WEOF 9TA80
:COPY (FILE,D1,XYZ),(OUT,9TA80)
!RADEDIT
!REWIND 9TA80
:COPY (FILE, D1, XYZ),(OUT,CPA04)
!EOD
Data Deck
:COPY (IN, CRA03), (FILE, D1, XYZ)
:(FORMAT, B), (SIZE, 500)
:ALLOT (FILE, D1, XYZ), (RSIZE, 20);
!RADEDIT
!JOB

Figure 9. RAD Editor COPY Example

interrupt and keys in one of the following commands to the Monitor:

C	Continue and read next record from the C device.
X	Abort RAD Editor and return control to Monitor.
COC	Continue and read a record from the OC device (used only in conjunction with the error message 'ERROR ITEM xx').

If the Editor aborts because of an irrecoverable I/O error, the physical device name is included in the abort message.

The error messages output by the RAD Editor and their meanings are given in Table 9.

RAD RESTORATION MESSAGES

The messages itemized in Table 10 are written on the keyboard/printer during RAD restoration via the bootstrap loader produced by SAVE. Unless otherwise specified, the computer will go into a WAIT after writing a message.

Table 9. RAD Editor Error and Status Messages

Message	Meaning	Action Taken
ERROR ITEM xx	Item number xx on the command is in error.	If operator response is C, Editor reads next record from C device. If operator response is COC, the next record is read from the OC device. This will enable operator to rectify a command error.
ILLEGAL BINARY RECORD	An illegal binary record (first byte not X'1C', X'3C') has been read with an object module.	If operator response is C, Editor reads next record from specified device.
CKSM ERROR	Last record in the object module being read has a checksum error.	If operator response is C, Editor reads next record from specified device.
SEQ ERROR	Last record in the object module being read has a sequence error.	If operator response is C, Editor reads next record from specified device.
EOT on $\begin{cases}\text{yyndd}\\\text{area,name}\end{cases}$	Unexpected end-of-tape was encountered on the specified device or file.	Operation is aborted.
yyndd WRT PROT	Specified RAD is write-protected.	Interrupt and key-in "SYC", or reset appropriate RAD protection switches. Or, if job is not allowed to write on protected areas of RAD, interrupt and key-in "X" to abort.
RAD OVERFLOW	Allocating the amount of RAD storage indicated by the "file" parameter on the :ALLOT command would cause the permanent RAD area indicated by the "directory" parameter to overflow.	Operation is aborted.
INVALID RSIZE. UNBLOCKED ORGANIZATION GIVEN	Maximum record size for a blocked file has been exceeded. Unblocked organization given.	Editor continues.
AREA xx IS NOT ALLOCATED	Specified area was not allocated at SYSGEN	Operation is aborted.
KEY ERR	Operator key-in is erroneous.	Key-in has to be either C, COC, or X.
SPECIFIED FILE DOES NOT EXIST	File does not exist within the specified area.	Operation is aborted.
DUPLICATE FILE	An attempt has been made to allocate a file using a name which already exists.	Operation is aborted.

Table 9. RAD Editor Error and Status Messages (cont.)

Message	Meaning	Action Taken
ILLEGAL FILE NAME	An attempt has been made to allocate a file using GO, OV, or X1-X9 as a file name.	Operation is aborted.
AREA xx CANNOT CONTAIN A RESIDENT FOREGROUND PROGRAM	Illegal area specified. Only the FP area can contain a resident foreground program.	Operation is aborted.
AREA SPECIFIED DOES NOT CONTAIN A LIBRARY	An area other than SP or FP was specified that does not contain a library.	Operation is aborted.
TRACK xxxxx CANNOT BE DELETED	Illegal attempt to remove a track from use containing a sector of the file directory. Removal would prevent accessing of files or other sectors of the directory.	Operation is aborted.
SPECIFIED ROM DOES NOT EXIST	Relocatable Object Module (ROM) does not exist within the specified library.	Operation is aborted.
REFERENCES TO F4:COM NOT ALLOWED	An external definition or reference F4:COM encountered in a Relocatable Object Module (ROM) being copied to the library.	RAD Editor skips to the end of the module. A key-in of C causes Editor to read next record from specified device.
ROM DOES NOT CONTAIN A DEF	Relocatable Object Module (ROM) being copied does not contain an external definition.	A key-in of C causes Editor to read next record from specified device.
DUPLICATE DEF xxxxxxxx	Relocatable Object Module (ROM) being copied to the library contains duplicate definitions.	RAD Editor skips to the end of the module. A key-in of C causes Editor to read next record from specified device.
ILLIGAL LOAD ITEM xx	Relocatable Object Module (ROM) to the library contains an illegal load item.	RAD Editor skips to the end of the module. A key-in of C causes Editor to read next record from specified device.
FILE xxxxxxxx WAS NOT TRUNCATED. FSIZE =0	File was not truncated because the file size being 0 suggests either a direct access file or a file with 0 records.	Editor continues.
SREC VALUE GREATER THAN EREC VALUE	Parameter error on the :DUMP directive. The last record to be dumped precedes the initial area to be dumped.	Operation is aborted.
AREA xx CONTAINS NO FILES	Specified area contains no files.	Editor continues.
RECORD SIZES DIFFER ON INPUT AND OUTPUT FILES	Record sizes differ on copying from RAD file to RAD file.	Operation is aborted.
ILLEGAL OPTION xxx	Option specified is not permitted on a :COPY command.	Operation is aborted.
BUFFER SMALLER THAN DATA READ	Data read exceeds available buffer space.	Operation is aborted.
NOT ENUF BACKG SPACE	Insufficient background space to perform requested operation.	Operation is aborted.

Table 9. RAD Editor Error and Status Messages (cont.)

Message	Meaning	Action Taken
UNABLE TO FIND AREA xx	Specified area cannot be found on RAD SAVE tape during a :RESTORE operation.	Operation is aborted.
AREA xx INCOMPATIBILITY	Attempting to restore specified area onto a different type of RAD from which it was saved, or the area to be restored is too large for the same area using the current Master Directory.	Operation is aborted.
AREA xx CKSM ERROR	A checksum error exists on the RAD SAVE tape in the specified area.	Operation is aborted
AREA xx TRUNCATED	Specified area being restored is larger than the same area using the current Master Directory, but the data that was lost contained all zeros.	Operation continues.
SAVE TAPE OK	RAD SAVE tape has been verified correctly.	No action.
CKSM ERR ON SAVE TAPE	A checksum error has been encountered while verifying the RAD SAVE tape.	Operation is aborted.
AREA SPECIFIED IS NOT MAINTAINED BY THE RAD EDITOR	An attempt has been made to use area CK, XA, or BT which is not maintained by the Editor.	Operation is aborted.
ILLEGAL USE OF :COPY	The specified combination of input and output devices on the :COPY command is prohibited.	Operation is aborted.

Table 10. RAD Restoration Messages

Message	Meaning	Resulting Action
yyndd WRT PROT	RAD is write-protected	Program will attempt the RAD write after an SY key-in.
CKSM ERROR	A checksum error has occurred in reading the SAVE tape	If WAIT condition is cleared, bootstrap loader continues and accepts bad record.
RAD RESTORED OK	RAD restoration has been successfully completed.	Control is transferred to the RAD bootstrap.
yyndd ERROR, SB=xxxx	A parity or transmission error has occurred on device yyndd. The device status byte (SB=) is also displayed.	There is no recovery.
yyndd UNUS. END, TDV=xxxx	An unusual end status has been returned from the specified device. The TDV status byte (TDV=) is also displayed.	There is no recovery on a read operation. On a write operation, the write is tried again after WAIT is cleared.

Table 10. RAD Restoration Messages (cont.)

Message	Meaning	Resulting Action

TRK=xxxx
DATA=ALL ZEROS
	Specifies contents of RAD controller address register in hexadecimal at the time of a check write error.	If the data being written contains all zeros, this information is output. If WAIT condition is cleared, bootstrap loader continues.

yyndd UNRECOG., SB=xxxx	An unrecognized status has been returned from the indicated device. The device status byte is also displayed.	Upon clearing the WAIT condition, operation is retried.

7. OVERLAY LOADER OPERATIONS

The Overlay Loader creates programs for execution in overlay format to reduce core space requirements during execution. An overlay program consists of a root and any number or segments.

OPERATOR HANDLING

Executable versions of programs with overlay segments are identical to nonoverlayed programs in terms of operator intervention. The Loader uses two passes to create the segmented program. The only listed output from the Overlay Loader is a map and possible diagnostic messages.

ERROR DIAGNOSTICS

The Overlay Loader outputs diagnostic messages on the OC and LL devices. Duplication is suppressed if OC and LL are assigned to the same device. If an operator response is required, the Loader calls the Monitor WAIT function. The operator should initiate an INTERRUPT and key in one of the following:

C	Continue
X	Abort
COC	Read the corrected command from OC and continue (used only in response to control command errors).

Note that the Monitor WAIT routine aborts if an !ATTEND control command has not been encountered in the job stack.

The diagnostic messages listed in Table 11 are output by the Overlay Loader.

Table 11. Overlay Loader Diagnostics

Text	Meaning	Action
BACKGROUND TOO SMALL	User's program cannot be loaded in the current size of the background. This is a function of the number of external symbols and forward references that a program has, not a function of the program length.	Abort
BINARY CARD ENCOUNTERED INSTEAD OF CC	A binary record instead of a control command was encountered on the C device.	Wait
BOT ON $\begin{cases}\text{yyndd}\\ \text{area,name}\end{cases}$	Unexpected beginning-of-tape has been encountered on the specified device/file.	Abort
BUF SMALLER THAN DATA RECORD DCB x:xxxxxx	Specified DCB has been assigned to a record size larger than the I/O buffer associated with the Read request. Either user has assigned incorrectly, or Loader has a program error.	Abort
CC ERR: DUP NAME IN ITEM xx	Item number xx on the command is a duplicate of a name in the symbol table.	Wait
CC ERR: DUP SEG IDENT	Ident on SEG command is a duplicate of a previous segment's ident.	Wait
CC ERR: FOLLOWING ITEM xx	There is an error following item xx on the command (e.g., a parameter has been omitted, an extra parameter has been included, etc.).	Abort if OLOAD CC. Wait if any other CC.

Table 11. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
CC ERR: ILLEGAL CC SEQUENCE	Loader commands have been ordered incorrectly.	Wait
CC ERR: ILLEGAL OPTION FOR PUBLIB LOAD (PUBL,name)	Option (PUBLIB,name) was specified on OLOAD and a PUBLIB command has been encountered.	Abort
CC ERR: ILLEGAL OPTION FOR PUBLIB LOAD (TASKS,value)	Option (TASKS,value) was specified on OLOAD and a PUBLIB command has been encountered.	Abort
CC ERR: ILLEGAL OPTION FOR PUBLIB LOAD (TEMP,value)	Option (TEMP,value) was specified on OLOAD and a PUBLIB command has been encountered.	Abort
CC ERR: ILL OPCODE IN ITEM xx	Specified operation code is either illegal or unimplemented.	Wait
CC ERR: ILL SEG IDENT	Seg ident on the MODIFY command does not match the segment being modified.	Wait
CC ERR: ITEM xx	Item number xx on the command is in error.	Abort if OLOAD CC. Wait if any other CC.
CC ERR: MODIFY OUTSIDE SEG LIMITS	One of the locations on the MODIFY command is outside the limits of the segment.	Wait
CC ERR: NEED (FORE,fwa,lwa) OPTION FOR PUBLIB LOAD	Option (FORE,fwa,lwa) was not specified on the OLOAD command and a PUBLIB command has been encountered.	Abort
CC ERR: SEG IDENT NOT 1ST OPTION	Segment ident (LINK,ident$_1$) is not the first option on the SEG command.	Wait
CC ERR: SEGMENTS ORDERED INCORRECTLY	SEG commands have been input in the wrong order.	Wait
CC ERR: SPECIFIED AREA FOR PUBLIB LOAD NOT 'FP'	Option (FILE,area,name) on OLOAD did not specify the Foreground Programs area (FP) and a PUBLIB command has been encountered.	Abort
CC ERR: STEP OPTION ILLEGAL WITHOUT ATTEND	An ATTEND command must be included in the job when the STEP option is specified.	Abort
CC ERR: UNDEFINED FILE,area,name	Area and file specified in the option (FILE,area,name) has not been defined by the RAD Editor.	Abort if OLOAD CC. Wait if any other CC.
CC ERR: UNDEFINED SYMBOL IN ITEM xx	Symbol name in item xx on the MODIFY command has not been defined.	Wait

Table 11. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
DCB CANNOT BE A DSECT SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\ \text{ROM}\\ \text{SLIB}\end{Bmatrix}$ xxx	A DCB was encountered in the named segment that was assembled as a dummy section instead of being part of the control section.	Abort
DCB HAS BAD PARAMETERS DCB x:xxxxxx	Specified DCB has bad parameters. Either user has assigned incorrectly, or Overlay Loader has a program error.	Abort
DCB HAS INSUFFICIENT INFO DCB x:xxxxxx	Specified DCB contains insufficient information to open a Read or Write operation. Either user has assigned incorrectly, or Loader has a program error.	Abort
DCB NOT ASSIGNED DCB x:xxxxxx	Specified DCB has been assigned to the "null" device. Either user has assigned incorrectly, or Overlay Loader has a program error.	Abort
DEFAULT ENTRY ADDRxxxxx SUPPLIED FOR ROOT	A transfer address was not encountered on any ROM in the root and an entry address was not specified on the CC; therefore, a default has been supplied.	Continue
DSECTs in PUBLIB LOAD SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\ \text{ROM}\\ \text{SLIB}\end{Bmatrix}$ xxx	Labeled COMMON blocks (DSECTs) are illegal in the Public Libraries.	Abort
EOT ON $\begin{Bmatrix}\text{yyndd}\\ \text{area,name}\end{Bmatrix}$	End-of-tape has been encountered on the specified device/file.	Wait
EXLOC TOO LARGE SEGxxxxx	Specified segment will exceed 131K at the given EXLOC.	Abort
FILE DESTROYED area,name	Overlay Loader is aborting past the point where data has been written on the specified program file. The first sector of the file has been zeroed out.	Abort
FILE UNCHANGED area,name	Overlay Loader is aborting at a point where the program file is unchanged.	Abort
ILLEGAL LOAD LOCATION xxxxx SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\ \text{ROM}\\ \text{SLIB}\end{Bmatrix}$ xxx	Specified "load location" origin has been defined with a value that is either not an address or that lies outside the address limits of the specified segment.	Abort
ILL SEG IDENT TERMINATED MODIFY'S	MODIFY commands have been ordered incorrectly for the (GO,LINKS) option. The MODIFY mode has been terminated at this point. If user wishes to continue, all MODIFY commands that follow will be ignored.	Wait

Table 11. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
INITIALIZING LCOM OUTSIDE SEG SEGxxxxx	A labeled COMMON block must be initialized in the segment where the block is allocated.	Continue (the labeled COMMON block will not be initialized).
LIB ROM'S EXCEED MAX SEGxxxxx	Maximum number of library ROMs that can be loaded is 2000.	Abort
MONITOR CC ENCOUNTERED INSTEAD OF :ROOT or :PUBLIB	Monitor control command instead of a ROOT or PUBLIB command was encountered on the C device.	Abort
MOUNT PAPER TAPE ROM	STEP option was specified on OLOAD and the next Relocatable Object Module (ROM) is to be input from the paper tape reader. Operator should load the paper tape, interrupt, and key in "C".	Wait
PROGRAM ERR: $\left\{\begin{array}{l}\text{CCI}\\\text{ONE}\\\text{TWO}\\\text{MAP}\\\text{LIB}\end{array}\right\}$ ADDRxxxx SEGxxxxx $\left\{\begin{array}{l}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{array}\right\}$ xxx	Loader has a program error in the named overlay at the specified address.	Abort. Operator should get a core dump.
PROGRAM ERR: $\left\{\begin{array}{l}\text{CCI}\\\text{ONE}\\\text{TWO}\\\text{MAP}\\\text{LIB}\end{array}\right\}$ SB = xx,ADDRxxxx DCB x:xxxxxx	Specified error status has been returned from an Overlay Loader call (in the named overlay) to a Monitor I/O routine. The address of the CAL and the name of the DCB are specified.	Abort
PROGRAM ERR: UNALLOCATED CSECT SEGxxxxx $\left\{\begin{array}{l}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{array}\right\}$ xxx	Loader has encountered a control section that has not been allocated; either a Loader, compiler, or assembler error.	Abort
RAD FILE TABLE FULL	RAD File Table size that was allocated at SYSGEN is insufficient.	Abort
READING AN OUTPUT DEVICE DCB x:xxxxxx	A DCB that the Overlay Loader reads with has been assigned to an OUT device. Either user has assigned incorrectly, or Loader has a program error.	Abort

Table 11. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
ROM ERR: BAD SEQ SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{Bmatrix}$ xxx SEQNOxxx	Sequence number of the binary record does not equal xxx.	Wait
ROM ERR: CHKSUM SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{Bmatrix}$ xxx SEQNOxxx	Specified binary record has a checksum error.	Wait
ROM ERR: EXPRESSION SIZE EXCEEDS MAX SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{Bmatrix}$ xxx SEQNOxxx	An object language expression on the specified binary record exceeds 120 bytes.	Abort
ROM ERR: ILLEGAL LOAD ITEM SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{Bmatrix}$ xxx SEQNOxxx	Object language on specified binary record cannot be translated (assembler or compiler error).	Abort
ROM ERR: NO MODULE END SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{Bmatrix}$ xxx SEQNOxxx	Module end was not encountered on the last binary record of the relocatable object module.	Abort
ROM ERR: NOT OBJECT LANGUAGE SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{Bmatrix}$ xxx SEQNOxxx	Specified binary record is not in object language format.	Wait
ROM ERR: NOT STANDARD BIN SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{Bmatrix}$ xxx SEQNOxxx	Specified record has a nonstandard binary format.	Wait
UNDEFINED FILE area,name DCB x:xxxxxx	Specified DCB has been assigned to a RAD file that has not been defined by the RAD Editor.	Abort
UNDEFINED ORIGIN SEGxxxxx $\begin{Bmatrix}\text{ULIB}\\\text{ROM}\\\text{SLIB}\end{Bmatrix}$xxx	Loader has encountered a "load location" origin with an expression that cannot be resolved.	Abort
UNEXPECTED EOD ON $\begin{cases}\text{yyndd}\\\text{area,name}\end{cases}$	Unexpected !EOD was encountered on the specified device/file.	Wait if the EOD was encountered instead of a ROM; otherwise, Abort.

Table 11. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
UNEXPECTED MONITOR CC ON $\left\{\begin{array}{l}\text{yyndd}\\ \text{area,name}\end{array}\right.$	Unexpected Monitor control command was encountered while reading ROMs from the C device.	Abort
UNRECOVERABLE RD ERR ON $\left\{\begin{array}{l}\text{yyndd}\\ \text{area,name}\end{array}\right.$	Transmission error has occurred while reading from the specified device/file.	Abort
UNRECOVERABLE WR ERR ON $\left\{\begin{array}{l}\text{yyndd}\\ \text{area,name}\end{array}\right.$	Transmission error has occurred while writing on the specified device/file.	Abort
WARNING: DCB IN OVERLAY SEGMENT SEGxxxxx $\left\{\begin{array}{l}\text{ULIB}\\ \text{ROM}\\ \text{SLIB}\end{array}\right\}$ xxx SEQNOxxx DCB x:xxxxxx	Specified DCB was declared an external DEF in a segment other than the ROOT. The DCB will not be included in DCBTAB.	Continue
WARNING: DEF'D DCB NOT DEFINED DCB x:xxxxxx	Specified DCB was declared an external DEF and the DEF was never defined.	Continue
WARNING: DUPLICATE DEF'S	User's program contains duplicated external DEFs. Map will indicate the name(s) of the DEFs.	Continue
WARNING: DUPLICATE REF'S	User's program contains duplicate external REFs. Map will indicate the name(s) of the REFs. Occurs when identical DEFs in different segments of different paths are referenced by the same REF (in a segment common to both paths).	Continue
WARNING: ENTRY ADDRxxxxx OUTSIDE SEGMENT SEGxxxxx	Entry address for specified segment is outside segment's address limits.	Continue
WARNING: ILLEGAL DCB ADDR DCB x:xxxxxx	Specified DCB was declared an external DEF and the DEF has been defined with either a negative address or a constant.	Continue
WARNING: ILLEGAL DCB NAME SEGxxxxx $\left\{\begin{array}{l}\text{ULIB}\\ \text{ROM}\\ \text{SLIB}\end{array}\right\}$ xxx SEQNOxxx DCB x:xxxxxx	Specified DCB name is illegal and will not be included in DCBTAB. Monitor DCBs (M:) must have standard OPLB names. User DCBs (F:) must not exceed eight EBCDIC characters in length.	Continue
WARNING: LCOM name OF SIZExxxx GREATER THAN ALLOCATED SEGxxxxx $\left\{\begin{array}{l}\text{ULIB}\\ \text{ROM}\\ \text{SLIB}\end{array}\right\}$ xxx SEQNOxxx	Named labeled COMMON blocks (DSECT) with the size specified (words) is greater than the size allocated.	Continue
WARNING: NO ENTRY ADDRESS FOR ROOT	Root does not have an entry address.	Continue

Table 11. Overlay Loader Diagnostics (cont.)

Text	Meaning	Action
WARNING: OVERLAY SEG GREATER THAN 16K SEGxxxxx	Specified overlay segment exceeds the maximum size record that can be loaded by the Monitor SEGLOAD function.	Continue
WARNING: PROGRAM EXCEEDS SPECIFIED ADDR LIMITS	User's program exceeds address limits, either those specified on OLOAD or the defaults for background/foreground programs.	Continue
WARNING: UNDEFINED DEF'S	User's program contains external DEFs that either have not been defined or have been defined with an expression the Loader cannot resolve. Map will indicate the name(s) of the undefined DEFs.	Continue
WARNING: UNDEFINED ENTRY ADDR SEGxxxxx	Expression defining entry address for specified segment cannot be resolved by Loader.	Continue
WARNING: UNSATISFIED REF'S	User's program contains unsatisfied external REFs. Map will indicate the name(s) of the REFs.	Continue
WRITING ON INPUT DEVICE DCB x:xxxxxx	A DCB that the Overlay Loader writes with has been assigned to an IN device. Either user has assigned incorrectly, or Loader has a program error.	Abort
yyndd WRITE PROT	Specified RAD is write-protected.	Wait and 1. Reset RAD protection switches, or 2. Interrupt and key-in "SYC", or 3. Interrupt and key-in "X" if the job is not allowed to write on protected areas of the RAD.

OVERLAY CONTROL COMMANDS

When the !OLOAD control command is read by the Job Control processor, it causes the Overlay Loader processor to be read into the background and executed. All Loader subcommands are identified by a leading colon (e.g., :SEG). They are read from the C device and logged onto LL. Blank cards are passed over without comment. When the next Monitor control command is encountered, the Loader completes the load process and exits to the Monitor.

!EOD can only be used as a terminator for object module input; its use is illegal for terminating the Loader Control command stack.

CONTROL COMMAND SEQUENCE

The control command stack is divided into major divisions or substacks, which must occur in the order given in Figure 10.

In Figure 10, the :COMMON, :LCOMMON, :LIB, :INCLUDE, :EXCLUDE, :RES, and :MODIFY commands may occur in any :ROOT or :SEG substack and apply only to that root or segment. The :ASSIGN commands must follow all other commands in the stack.

ROOT OR SEG SUBSTACK SEQUENCE

A ROOT or SEG substack has the order given in Figure 11.

PUBLIB SUBSTACK

The :PUBLIB substack may be used to replace the :ROOT, :SEG, and :ASSIGN substacks within the control command stack. In this case, the :PUBLIB substack will have the order illustrated in Figure 12.

LOADER COMMAND FORMATS

All acceptable Loader commands are given in Table 12 and are listed in a logical, but not necessarily typical, operating sequence. Sample parameters are given in the "Example" column only to illustrate typical parameter formats. An explanation of the parameters is not given (a more detailed description is given in the XDS Sigma 5/7 RBM Reference Manual).

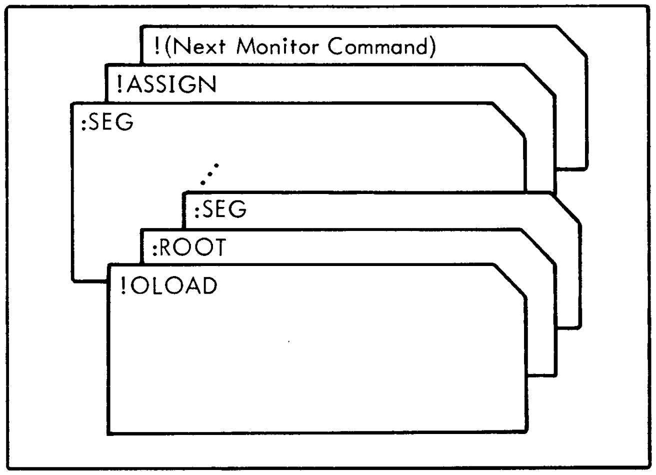
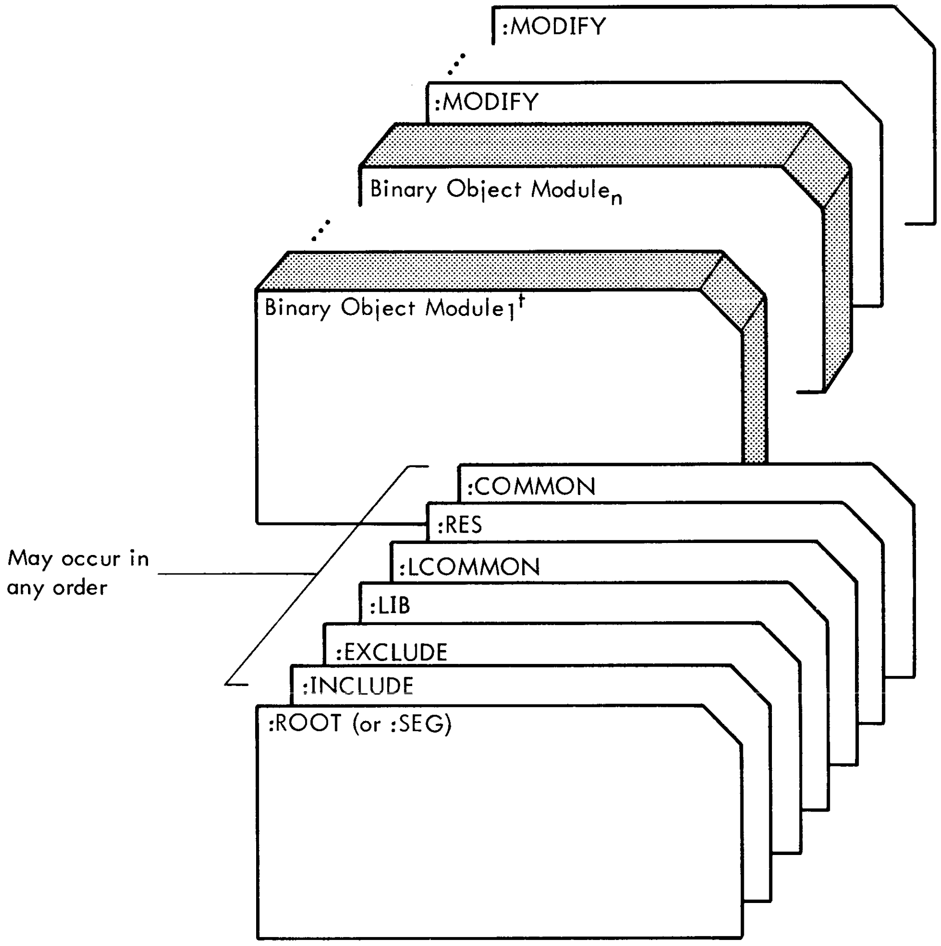


Figure 10. Major Substack Sequence



†Binary object modules are included at this point in the substack only if the input device specified on the preceding ROOT or SEG command is the same as the C device.

Figure 11. ROOT or SEG Substack Sequence

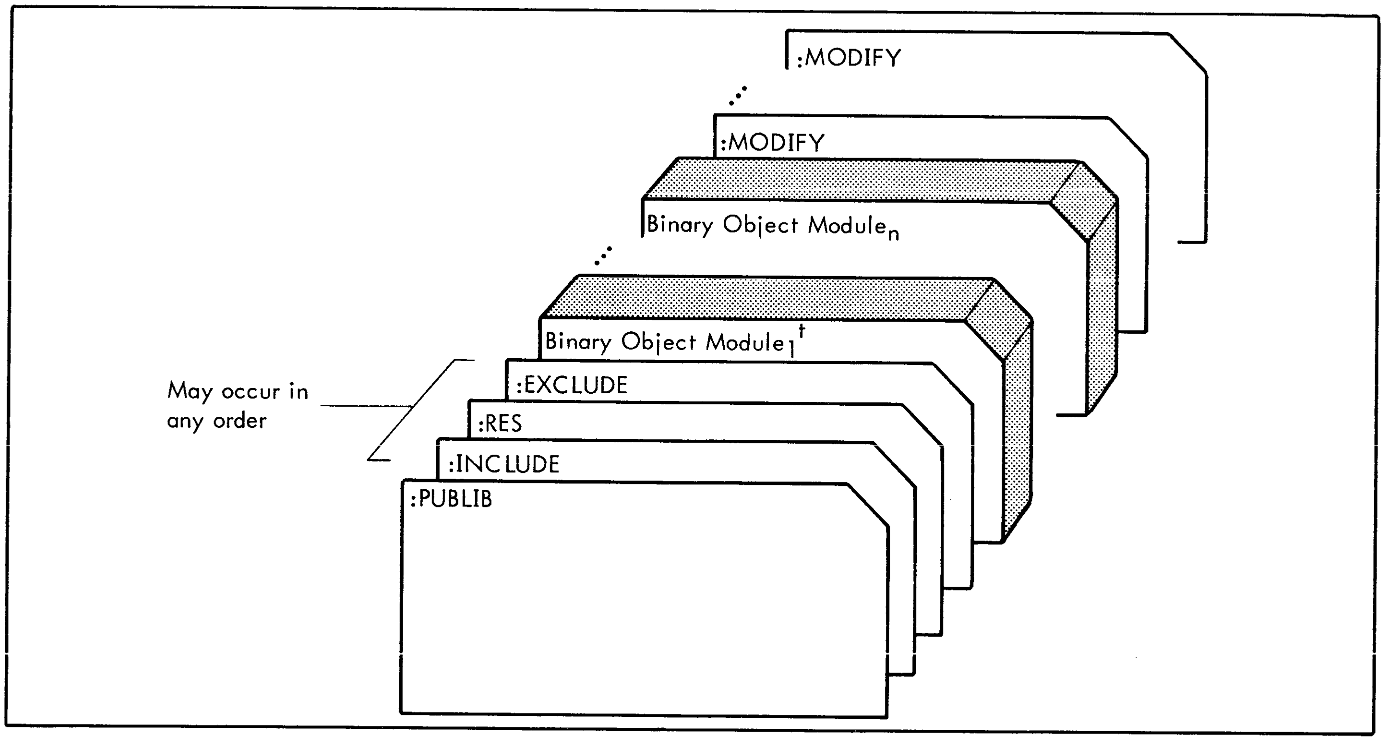


Figure 12. PUBLIB Sequence

Table 12. Loader Control Commands

General Form	Example	Meaning
!OLOAD [(option$_1$)(,option$_2$). . . (,option$_n$)]	!OLOAD (MAP,ALL), (FILE,BP,CALCLOAD)	Calls the Overlay Loader. Any error on the command causes Loader to abort. Recovery consists of correcting error and reloading entire job.
:ROOT [(ENTRY,def)(,input option$_1$),. . . (,input option$_n$)]	:ROOT (ENTRY,INIT), (DEVICE,CRA03)	Specifies object modules from which root segment is to be created. Must precede all SEG commands.
:SEG (LINK,ident$_1$ [,ONTO,ident$_2$), (EXLOC,addr), (ENTRY, def), (input option$_1$),. . ., (input option$_n$)]	:SEG (LINK,1,ONTO,0)	Defines a segment's overlay linkage and specifies object modules from which the segment is to be created.
:LIB [(USER,SYSTEM)]	:LIB (USER)	Specifies library search for one segment only (identified by the preceding ROOT or SEG command). Option keywords denote libraries and order of search. If neither is specified, library search (except for Public Library) is suppressed for that segment.
!INCLUDE (def$_1$ [,def$_2$,. . .,def$_n$])	:INCLUDE (9CADD, 8EDITT)	Permits routines to be loaded from libraries when no reference to the routine has been made in any module of the segment.

Table 12. Loader Control Commands (cont.)

General Form	Example	Meaning
:EXCLUDE (def$_1$ [,def$_2$, . . . ,def$_n$])	:EXCLUDE (8END10L, 6DATLINK)	Inhibits library search and linkage for the named definition(s) even though an external reference occurs in a module of the segment.
:COMMON	:COMMON	Sets the base of Blank COMMON at end of segment identified by the preceding ROOT or SEG command. If the command is not included, Blank COMMON is set at end of the longest path. Only one COMMON command can be present in the control command stack. Specification field must be blank.
:RES (def,size)$_1$ [,(def,size)$_2$, . . ., (def,size)$_n$]	:RES (RESA,200),(X,147)	Permits user to reserve and name one or more areas at the end of the segment for load-time or runtime debug purposes. Size is a decimal value.
:LCOMMON (blockname,size)$_1$ [, . . . ,(blockname, size)$_n$]	:LCOMMON (LCOMA, 1200,(QCOM,720)	Permits user to determine allocation of labeled COMMON blocks within root and overlay segments.
:MODIFY [(SEG,ident),] (LOC,address) ,word$_1$, . . . [,word$_n$]	:MODIFY (LOC,MAP+.FO) ,(B PATCH+6)	Modifies core locations of relocatable programs at load time. MODIFY commands must be input at the end of the ROOT, SEG, or PUBLIB substack for segment being modified.
:ASSIGN (dcb [,area,name]) [,(option),(option),. . .,(option)]	:ASSIGN (MLO,9TA81),VFC	Creates, initializes, or modifies DCBs at load time. Must be last commands in the control command stack.
:PUBLIB [(input option) , (input option) , . . . , (input option)]	1. :PUBLIB 2. :PUBLIB (FILE,FP,X), (DEVICE,9TA82,2), (FILE,BJ,GO,EOD)	Specifies the object modules from which the Public Library is to be created. Order of the parameters determines the order of loading.

OVERLAY LOADER DECK SETUPS

Examples of typical programs using the Overlay Loader are illustrated in Figures 13 and 14.

In Figure 13, the JOB card rewinds the GO file, the FORTRAN source deck is compiled, and the binary object module is output on GO. The Macro-Symbol compressed source deck is updated and the binary object module is output to file CALC2 in the D5 area (previously allocated by the RAD Editor). The ROM (Relocatable Object Module) implied on the :ROOT and designated on the :SEG commands is loaded, and the loaded program is output to CALCLOAD in the BP area. The :ROOT command causes the ROM created by FORTRANH to be loaded from the GO file and creates the Root. The ROMs following the first :SEG command are loaded until !EOD is encountered and segment 1 is then created. The next :SEG command loads the ROM assembled by Macro-Symbol on the CALC2 file in the D5 area and creates segment 2. The !RUN command executes the loaded segmented program.

In Figure 14, the GO file is rewound by the initial !JOB command for the first FORTRAN compilation. The Overlay Loader loads from the GO file to form a root and outputs on the OV file for execution. A map is not output since MAP option is not specified. A postmortem dump is requested if the background aborts. The next !JOB command rewinds the GO file and three FORTRAN jobs are compiled, with the binary object modules output on GO to form ROM1, ROM2, ROM3 (Relocatable Object Modules). The Overlay Loader loads the first ROM for the root, the second ROM for segment 1, and the third ROM for segment 2. Note that :SEG cards are not required. The programs are executed from the OV file. A postmortem dump is specified in case an abort occurs.

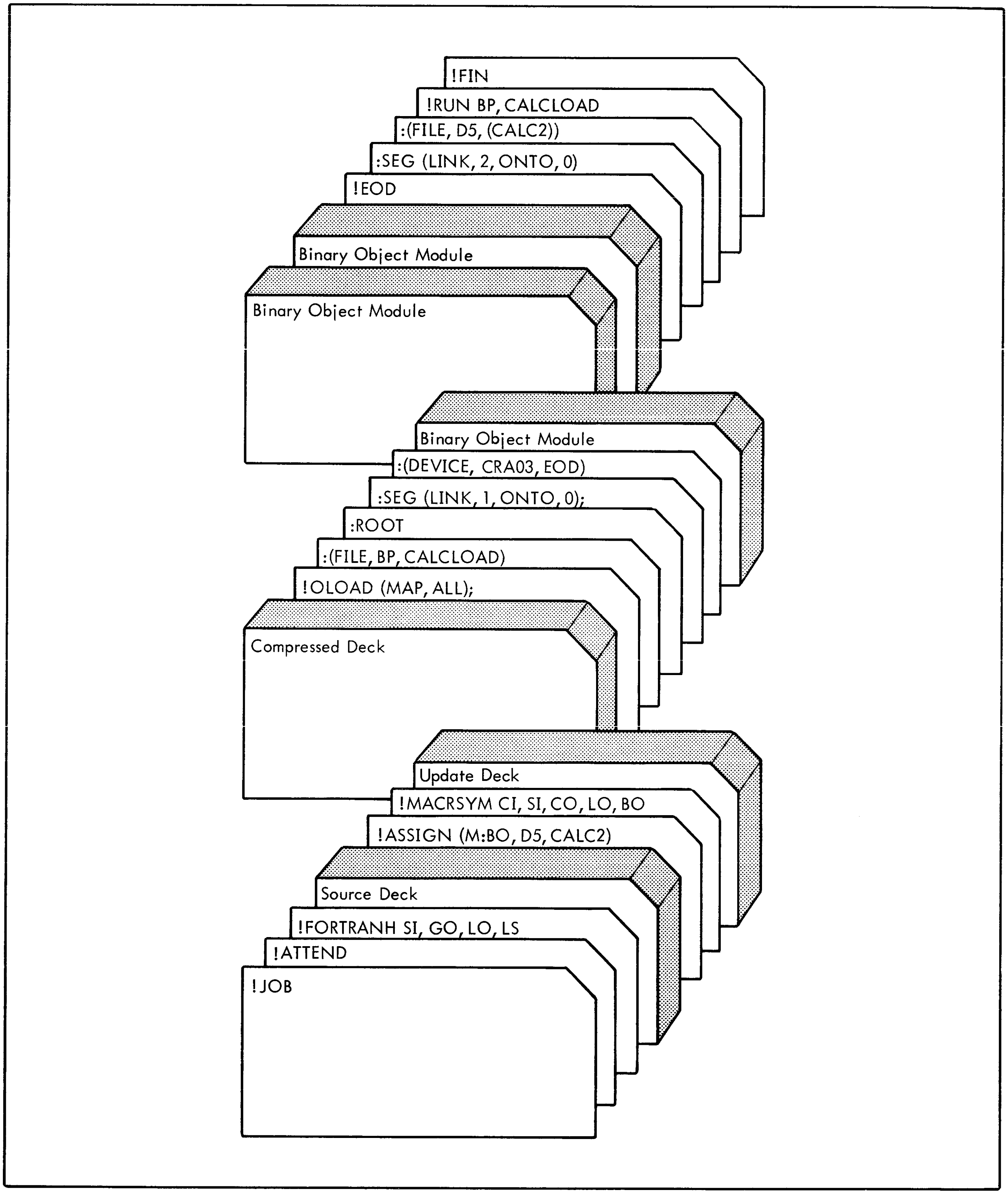


Figure 13. Overlay Loader Segmented Job Example

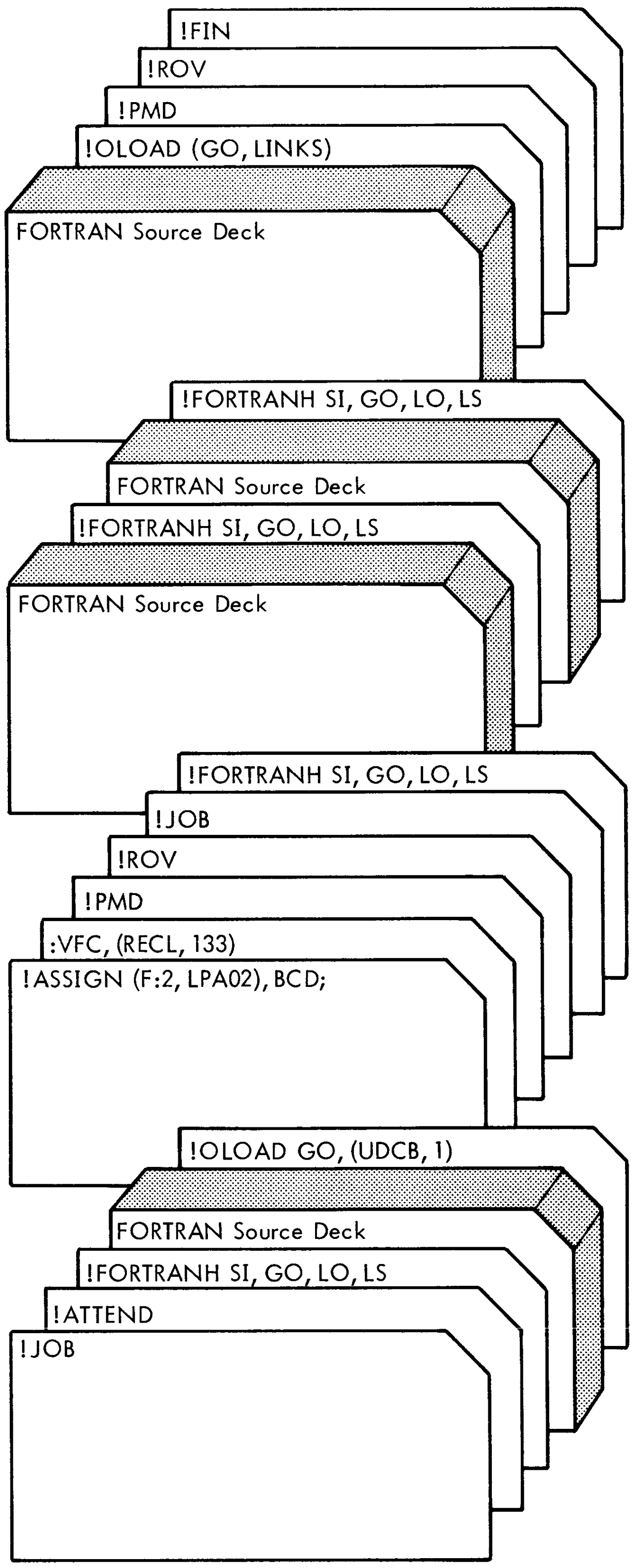


Figure 14. Overlay Loader Batch Example with GOLINKS