IBM

Series/1

SC34-0646-0

Event Driven Executive Installation and System Generation Guide

Version 5.0

Library Guide and Common Index SC34-0645	Installation and System Generation Guide SC34-0646	Operator Commands and Utilities Reference SC34-0644
Language Reference SC34-0643	Communications Guide SC34-0638	Messages and Codes SC34-0636
Operation Guide SC34-0642	Event Driven Language Programming Guide SC34-0637	Reference Cards SBOF-1625
Problem Determination Guide SC34-0639	Customization Guide SC34-0635	Internal Design LY34-0354

IBM

Series/1

SC34-0646-0

Event Driven Executive Installation and System Generation Guide

Version 5.0

Library Guide and Common Index SC34-0645	Installation and System Generation Guide SC34-0646	Operator Commands and Utilities Reference SC34-0644
Language Reference SC34-0643	Communications Guide SC34-0638	Messages and Codes SC34-0636
Operation Guide SC34-0642	Event Driven Language Programming Guide SC34-0637	Reference Cards SBOF-1625
Problem Determination Guide SC34-0639	Customization Guide SC34-0635	Internal Design LY34-0354

Fisrt Edition (December 1984)

Summary of Changes for Version 5.0

The following additions and changes have been made to this document:

- Chapter 3 has been updated to include the IBM product diskettes for Version 5.
- Chapter 4 has been updated to include the Remote Support Link requirements.
- Chapter 5 has been updated to reflect Version 5 changes to $EDXDEF and product diskettes.
- Chapter 6 has been updated to include migrating to Version 5.
- Appendix A has been updated to include the Remote Support Link requirements.
- Appendix D has been updated to include new supervisor module names.
- Miscellaneous editorial changes have been made also.

About This Book

This book is a guide for installing the Event Driven Executive on your Series/1 and for generating a tailored operating system to meet your application requirements. It contains step-by-step procedures for installation and system generation tasks.

Audience

This book is intended for anyone who has to install the Event Driven Executive on an IBM Series/1 and to create an operating system to meet application requirements. Readers should have a basic understanding of computer terminology before using this book.

How This Book Is Organized

This book is divided into the following 6 chapters and 5 appendices.

- *Chapter 1, Prepare for Installation and System Generation*, contains an overview of the topics covered in this book.
- *Chapter 2, Determine If the Starter System Meets Your Needs*, describes the I/O devices and software features supported by the starter system, an IBM-supplied operating system.
- *Chapter 3, Install EDX*, provides the step-by-step procedures for installing the Event Driven Executive system on your IBM Series/1.
- *Chapter 4, Select Your Required Support*, describes how to select the support (hardware and software features) you need for your application.

How This Book Is Organized *(continued)*

- *Chapter 5, Generate a Tailored Operator System*, provides the step-by-step procedures for generating a tailored operating system.
- *Chapter 6, Migrate to Version 5,* provides the migration procedures used to convert from EDX Versions 1 and 2 to EDX Version 5.
- *Appendix A, System Definition Statements*, contains the system definition statements used to define I/O devices to your operating system.
- *Appendix B, Customizing Adapters with Hardware Jumpers*, contains information about jumper connections on several adapters.
- *Appendix C, 3101 Configuration Information*, contains planning information to use in setting up and defining 3101 display terminals to your operating system.
- *Appendix D, Supervisor Module Names (CSECTS)*, contains a listing of all supervisor module names and entry point labels.
- *Appendix E, Work Sheets*, contains four work sheets. Use work sheet 1 to estimate your storage requirements and the partition structure of processor storage. Use work sheet 2 to define the processor storage characteristics and the I/O devices attached to your Series/1. Use work sheet 3 to define the software features you require to support the I/O devices you defined in work sheet 2 and the EDX-related products you include in your system. Use work sheet 4 to estimate the size of supervisor partitions and programs executing in partitions other than partition one so that you can define the structure of processor storage.

Aids to Using This Book

This book contains the following aids to using the information it presents:

- A table of contents that lists the major headings in this book.
- An index of the topics covered in this book.
- A glossary that defines terms and acronyms used in this book and in other EDX library publications.

In the step-by-step procedures shown in Chapters 3 and 5, several utilities are used and the interactive display screens are shown. Any responses you must make to a system prompt are shown in red.

A Guide to the Library

Refer to the EDX *Library Guide and Common Index* for information on the design and structure of the Event Driven Executive library, a bibliography of related publications, and an index to the entire library.

Contacting IBM about Problems

You can inform IBM of any inaccuracies or problems you find when using this book by completing and mailing the **Reader's Comment Form** provided in the back of the book.

If you have a problem with the Series/1 Event Driven Executive services, fill out an authorized program analysis report (APAR) form as described in the *IBM Series/1 Software Service Guide*, GC34-0099.

Contents

Contents

Figures

Chapter 1. Prepare for Installation and System Generation

This chapter outlines what you need to do to install the Event Driven Executive (EDX) or create a tailored operating system.

EDX is shipped to you on a series of product diskettes that contain a basic supervisor program and related support. The basic system supports certain hardware devices and software features. If the hardware devices and software features supported by this system match your application needs, you can install it as your operating system.

If you have hardware devices attached to your Series/1 that are not supported by the supplied system, or if you require software features that it does not include, you must generate a tailored operating system.

What Do You Need To Do?

Once you receive the Program Directory[1] and the product diskettes, you need to:

- Migrate to Version 5 from Versions 1 and 2 (Chapter 6).
- Determine if the supplied system meets your requirements (Chapter 2).
- Install EDX (Chapter 3).
- Select your required support (Chapter 4).
- Generate a tailored operating system (Chapter 5).

[1] The Program Directory is a set of installation procedures and instructions shipped with the product diskettes.

What Do You Need To Do? *(continued)*

Migrate to Version 5 (Version 1 and 2 Only)

Before installing the EDX Version 5, you must convert Versions 1 and 2 data sets to make them compatible with Version 5 data sets. The conversion procedures to migrate from Versions 1 and 2 are described in Chapter 6.

Note: If you are a current user of EDX Versions 3, 4.0, or 4.1 and are migrating to EDX Version 5, you do not have to perform a special conversion procedure. Versions 3, 4.0, 4.1, and PTF P02 programs and data sets are compatible with Version 5. However, we suggest that you create a backup copy of your current system and pertinent data before installing EDX Version 5.

Determine if the IBM-Supplied System Meets your Requirements

The information in Chapter 2 helps you decide if the IBM-supplied system meets your application needs. If it does, you can use it as your operating system. If it does not, you must generate a tailored operating system.

Install EDX

EDX is supplied to you on the product diskettes. To install EDX on your Series/1, you must copy these diskettes to your disk. The installation procedure varies, depending on the model processor you have on your system.

Chapter 3 describes the installation procedure for systems with a 4952, 4954, 4955, or 4956 processor. If you have one of these processors, you need to perform the following steps to install EDX:

- Migrate to Version 5 (Version 1 and 2 users only).
- IPL the supplied system from diskette.
- Allocate and initialize disk volumes.
- Copy the product diskettes to your disk.
- IPL the system from disk.
- Verify your installation using an IBM-supplied program.

What Do You Need To Do? *(continued)*

Select Your Required Support

The information in Chapter 4 helps you select the system features you need to generate a tailored operating system. This task consists of defining the structure of processor storage, defining the I/O devices attached to your Series/1, and selecting the system features required for your application.

The following work sheets are provided to help you keep track of your selection:

Work Sheet 1	Estimating the size of your supervisor
Work Sheet 2	Defining I/O devices to the supervisor
Work Sheet 3	Selecting software support
Work Sheet 4	Estimating the size of supervisor portions and programs executing in partitions other than partition one.

After you finish selecting the devices and features and filling out the work sheets, you can take these work sheets and this book to your Series/1. Then you can follow the procedures in Chapter 5 for generating a tailored operating system.

Generate a Tailored Operating System

Chapter 5 provides the steps you need to generate a tailored operating system. This means you must modify a copy of the EDX supervisor which is part of the IBM-supplied system. This process consists of the following tasks:

1. Creating a set of system definition statements reflecting the hardware configuration of the system on which the tailored operating system will run.
2. Selecting the supervisor object modules that are required to support the desired I/O devices and system features.
3. Assembling the system definition statements created in Step 1 above.
4. Link editing the object modules produced by the assembly in Step 3 with the supervisor object modules selected in Step 2 to produce a modified supervisor.
5. Initializing your new supervisor.
6. IPLing your system.
7. Verifying your operating system.

Notes

Chapter 2. Determine If the IBM-Supplied System Meets Your Needs

The information in this chapter helps you determine if the configuration of the IBM-supplied system meets your needs as an operating system. You can use this system configuration to define your operating system ***only*** if the devices you have and the software features you need match the devices and software that are built into the IBM-supplied system.

If you have any devices that aren't built into the IBM-supplied system (such as magnetic tape or two 4964 diskette units), you must "tailor" your operating system. You must also "tailor" your operating system if you need **any** software features that aren't built into the IBM-supplied system (such as floating point, error logging, spooling, or timers).

Tailoring your operating system means building your needs into an operating system. The task of tailoring your operating system involves adding I/O devices and software features that are not supported in the IBM-supplied system configuration as well as deleting from it the I/O devices and software features that you do not need. The system supplied on the product diskettes is one of the tools you need to tailor your own operating system. Therefore, you must install it even if you cannot use it as your operating system.

Determine If the IBM-Supplied System Meets Your Needs

What is the IBM-Supplied System?

The IBM-supplied operating system, $EDXNUC, supports up to 512K bytes of storage (depending upon your processor) and has eight partitions defined. A maximum of five programs can execute in each partition at the same time. In addition, the IBM-supplied system supports certain I/O devices and software features.

As received from IBM, $EDXNUC contains special supervisor modules that allow it to run on any Series/1 with the minimum system configuration. You should not use these modules in a production environment. Therefore, even if your hardware configuration is supported by the IBM-supplied system, you must generate a tailored system, using the supplied defaults.

Configurations for $EDXNUC

The following sections outline the hardware and software supported by the basic supervisor $EDXNUC.

Devices Supported by $EDXNUC

The starter system for the 4952, 4954, 4955, and 4956 processors ($EDXNUC) supports the following I/O devices:

One	4962, 4963, or 4967 disk storage unit
	or
One	30-megabyte disk within a 4952, 4954, or 4956 Model 30D processor or within a 4965 storage and I/O expansion unit (model 30D)
	or
One	60-megabyte disk within a 4954 or 4956 Model 60D processor or within a 4965 storage and I/O expansion unit (model 60D)
	plus
One	4964 diskette storage unit
Two	4965 diskette storage units
One	4966 diskette storage unit
One	4978, 4979, or 4980 Display Station
One	Teletypewriter[2] (TTY) device

[2] Trademark of Teletype Corporation

What is the IBM-Supplied System? *(continued)*

One	4974 printer
One	Multifunction Attachment (MFA) - Feature Number #1310
One	Multidrop Work Station Attachment - Feature Number #1250
One	4975-01L AND one 4975-02L printer attached via the Multifunction Attachment
One	3101 Display Terminal (block mode) locally attached at 1200 bits/second via the Multifunction Attachment.

Notes:

1. If there is no Multifunction Attachment, $EDXNUC expects a 3101 Display Terminal (block mode) attached point-to-point at 1200 bits per second and range high via adapter #1610. (For information on customizing adapters, see Appendix B, "Customizing Adapters with Hardware Jumpers" on page IS-243.)

2. If there is no #1610 adapter, $EDXNUC expects a 3101 Display Terminal (block mode) attached via adapter #2091/2092.

3. If there is no #2091/2092 adapter, $EDXNUC expects a 3101 display terminal (block mode) attached via adapter #2095/2096.

Software Features Supported by $EDXNUC

$EDXNUC provides the following software support:

- Initialization and operational support for supported devices
- Relocating program loader
- #1310, 2095/2096, 7850 ACCA/TTY translation
- #1610, 2091/2092 ACCA translation
- A 256-byte supervisor patch area
- MINMSG support.
- Error logging.

What is the IBM-Supplied System? *(continued)*

Devices Not Supported by $EDXNUC

The Series/1 devices in the following checklist are ***not*** supported by the basic starter system ($EDXNUC). Use the checklist below to indicate the I/O devices attached to your Series/1. If any of these devices is part of your system, you need to generate a tailored operating system.

Device	Yes	No
4968 Magnetic Tape Subsystem		
4969 Magnetic Tape Subsystem		
4973 printer		
5219 printer		
5224 printer		
5225 printer		
2741 Communications Terminal		
General Purpose Interface Bus		
Binary Synchronous Line		
Synchronous Communications Single-Line Control/High Speed Feature (#2080)		
Series/1-to-Series/1 Attachment (RPQ D02441 or D02242)		
Timer Feature #7840		
An ASCII terminal attached via feature #1610, #2091/2092, or #2096/2096		
4975-01A ASCII Printer		
Integrated I/O Nonisolated Feature (#1560)		
Printer Attachment - Series 5200 (#5640)		
4982 Sensor I/O Units		
More than two 4965 diskette units		
Virtual Terminal option		
Other I/O devices attached to the Series/1		

Figure 1. Checklist of Devices Not Supported by the Starter System ($EDXNUC)

You also need to generate a tailored operating system if your system contains ***multiples*** of the following devices:

Device	Yes	No
4962 disk		
4963 disk		
4967 disk		
30-megabyte disk		
60-megabyte disk		
4964 or 4966 diskette units		
4978, 4979, or 4980 display terminals		
3101 display terminal		
TTY		

Figure 2. Checklist of Devices Not Supported by the Starter System ($EDXNUC)

Software Features not Provided by $EDXNUC

The software features in the following checklist are ***not*** supported by the starter system (EDXNUC). Use the checklist to indicate the features you require for your application. If you do require any of these, you need to generate a tailored operating system.

Software Features	Yes	No
No overlay support		
Multiple wait support		
Queue processing support		
Binary synchronous communications support		
Timer support for the 4955 processor		
EXIO support		
EXIO trace option		
Unmapped storage support		

Figure 3 (Part 1 of 2). Checklist of Software Not Supported by EDXNUC

What is the IBM-Supplied System? *(continued)*

Software Features	Yes	No
Host communications support		
Minimum message support		
ACCA trace option		
Indexed Access Method QCBs		
Translation tables for 2741 terminal		
Timer support for the 4952/4954/4956 processor		
Software support modules for the I/O devices listed under "Devices Not Supported by $EDXNUC" on page IS-8		
1024 bytes/sector IPL capability and/or I/O support		
Interprogram communications via virtual terminals		
Spooling support		
EBCDIC/floating-point conversion		
Floating-point arithmetic		
Remote Management Utility		
Interactive debug		
Virtual Terminal option		

Figure 3 (Part 2 of 2). Checklist of Software Not Supported by EDXNUC

Chapter 3. Install EDX

This chapter describes two things. First, it describes what you need to install the Event Driven Executive (EDX) on a Series/1 with a 4952, 4954, 4955, or 4956 processor. Second, it describes the step-by-step procedures for installing EDX on one of these systems.

What Do You Need?

To install EDX, you need the following:

- The Program Directory
- The product diskettes shipped to you from IBM containing the following program library:
  - Basic Supervisor and Emulator (5719-XS5).
  - Any of the following program products depending on your installation requirements:
    - Program Preparation Facility (5719-XX6)
    - EDX Macro Library (5719-LM9)
    - EDX Macro Library/Host (5740-LM6).
- A Series/1 with one of the following minimum configurations, dependant on the processor installed on your system:
  - If you have a 4952, 4954, 4955 (except models A and C), or 4956 processor, you need the following minimum configuration:

What Do You Need? *(continued)*

One Processor (one of the following):

- 4952, 4954, 4955 (except models A and C), or 4956 processor with 64KB of storage for a production system

or

- 4952, 4954, 4955 (except models A and C), or 4956 processor with 128KB of storage for a development system.

One Disk storage unit (one of the following):

- 4962 Disk Storage Unit.
- 4963 Disk Subsystem.
- 4967 Disk Subsystem.
- 30-megabyte disk storage unit within a 4952, 4954, or 4956 model 30D processor or within a 4965 storage and I/O expansion unit (model 30D).
- 60-megabyte disk storage unit within a 4954 or 4956 model 60D processor or within a 4965 storage and I/O expansion unit (model 60D).

Note: The 30-megabyte disk is referred to as DDSK-30 and the 60-megabyte disk is referred to as DDSK-60.

One 4964 Diskette Unit, 4965 Diskette Drive, or 4966 Diskette Magazine Unit

One 4974 Printer or 4975 Printer (attached through the Multifunction Attachment (MFA) Feature #1310)

One Operator station (one of the following):

- 4978 or 4979 Display Station.
- 4980 Display Station attached through the Multidrop Work Station Attachment Feature #1250 at 100K bits per second.
- 3101 Display Terminal (character mode) or equivalent device attached through the Teletypewriter Adapter Feature #7850.
- 3101 Display Terminal (block mode) attached point-to-point through Feature #1610, #2091/2092, or #2095/2096 at 9600 bits per second and high speed range.
- 3101 Display Terminal (block mode) attached via a Multifunction Attachment Feature #1310 at 9600 bits per second in RS-422 mode.

What Do You Need? *(continued)*

In addition, these devices must be at the following specified addresses:

Device	Type	Address (Hex)
4967	Disk Subsystem	48
4964	Diskette Unit	02
4965	Diskette Drive	44[1]
4966	Diskette Magazine Unit	22
4974	Printer	01
4975-01L	Printer (via MFA)	5A
4975-02L	Printer (via MFA)	5B
4978	Display Station	24
4979	Display Station	04
4980	Display Station	80
#1250	Multidrop Work Station Attachment	80
#1610	Asynchronous Communications Single-Line Adapter	08[2]
#2091/2092	Asynchronous Communications Multiline Adapter	60[2]
#2095/2096	Feature-Programmable Communications Adapter	68[2]
#7850	Teletypewriter Attachment	00[3]
#1310	Multifunction Attachment (MFA)	58[4]

Figure 4. Address Assignments for Minimum Series/1 Configuration

Notes:

1. If the diskette unit is part of a 4952 model 30D processor, a 4954 or 4956 model 30D/60D processor, or a 4965 storage and I/O expansion unit (model 30D/60D), the diskette unit must be at address 45.

2. Supports 3101 models 20, 22, 23 in block mode.

3. Supports 3101 models 10, 12, 13, 20, 22, 23 in character mode.

4. Supports 3101 model 23 in block mode at address 59.

What Do You Need? *(continued)*

Preparing to Install EDX

Note: This discussion is limited to the Basic Supervisor and Emulator (5719-XS5) and the Program Preparation Facility (5719-XX6). The Macro Library (5719-LM9) and Macro Library/Host (5740-LM6) licensed programs are not addressed.

Before you install EDX, your system must have one of the following disk units:

- 4962 Model 1 or 2
- 4962 Model 1F or 2F
- 4962 Model 3 or 4
- 4963 Model 1
- 4963 Model 1F
- 4963 Model 2
- 4963 Model 2F
- 4967 Model 2
- 4967 Model 4
- DDSK-30
- DDSK-60.

An Event Driven Executive supervisor must reside in a data set named $EDXNUCx (where x equals any alphameric character you wish to specify). As shipped from IBM, diskette XS5001 contains a supervisor called the starter system in a data set named $EDXNUC. The first step in installation requires that you perform an IPL of the starter system from diskette XS5001. Therefore, the Series/1 must have one of the following devices wired as either the Primary or Alternate IPL device:

- 4964 Diskette Unit at hardware address X'02'
- 4965 Diskette Drive at hardware address X'44' or X'45'
- 4966 Diskette Magazine Unit at hardware address X'22'.

Notes:

1. If the diskette device is wired as the Primary IPL device, the disk device must be wired as the Alternate IPL device.

2. If a 4966 is the IPL device, you can perform an IPL from diskette slot 1 only.

The object of installation is to copy the programs and utilities supplied on the product diskettes to a disk device. Therefore, the Series/1 on which the starter system is being installed must have one of the following devices wired as the Primary or Alternate IPL device.

- 4962 disk (any model) at hardware address X'03'
- 4963 disk (any model) at hardware address X'48'
- 4967 disk (any model) at hardware address X'48'
- DDSK-30 disk at hardware address X'44'
- DDSK-60 disk at hardware address X'44'.

What Do You Need? *(continued)*

Note: If the disk device is wired as the Primary IPL device, the diskette device must be wired as the Alternate IPL device.

In addition, the starter system assumes that certain terminal devices are attached to the Series/1, and that they have specific address assignments. The Series/1 on which the starter system is being installed must have one of the following:

- TTY device at hardware address X'00'
- 4978 display at hardware address X'24'
- 4979 display at hardware address X'04'
- 4980 display at hardware address X'80'.

Also, the Series/1 may have a 4974 Printer at hardware address X'01', a 4975-01L at hardware address X'5A', or a 4975-02L at hardware address X'5B'. If you have a 4973, you can use the RA command of the $TERMUT1 utility to reassign $SYSPRTR as follows:

```
RA $SYSPRTR x
```

where x is the hardware address of your 4973.

Installing EDX

Note: The procedures and instructions for installing the Event Driven Executive are documented in the *Program Directory*, which is shipped with the licensed product diskettes. Late changes or updates to the installation process are reflected in the *Program Directory*. Therefore, you should use the *Program Directory* to install the EDX licensed programs. This discussion is limited to the major steps involved in installation.

To copy the product diskettes and install EDX, you must do the following:

Step 1 Save your existing system

Current users of EDX Versions 1 or 2 must perform a special migration procedure outlined in Chapter 6, "Migrate to Version 5" on page IS-117.

Current users of EDX Versions 3, 4.0, or 4.1 do not need to perform a special migration procedure. EDX Version 3, 4.0, or 4.1 programs are compatible with EDX Version 5. However, we suggest that you create a backup copy of your current system and pertinent data before installing EDX Version 5.

Installing EDX *(continued)*

The following are some things to consider when you are deciding whether to make a backup copy of your system:

- Are there any modified or special modules on the system such as:
  - A special $4978CS0 or $4980CSO on volume EDX002 for your own 4978 or 4980 PF key definitions?
  - A special $EDXL in ASMLIB for CF copy code, 370 Channel Attach, SNA, and so on.
  - Your own device initialization modules?
  - Modified EDX supervisor modules?
- Do you have any session manager parameters saved?
- Do you have a copy of your old $EDXDEFS, LINKCNTL, OR SUPPREPS?

Step 2 IPL the starter system from diskette XS5001.

Step 3 Initialize logical volumes EDX002, ASMLIB, EDX003, and FHVOL (for disks with fixed heads).

Note: If you are a current user of EDX Version 3 , 4.0 or 4.1, you may not need to initialize these volumes. They were initialized when you installed your current system and already exist. However, see "Step 3 - Initialize Logical Volumes" on page IS-21 for the recommended volume sizes. If your volumes are not large enough to contain the modules to be copied, you need to reinitialize them.

Step 4 Copy the starter system ($EDXNUC) and basic utilities.

Step 5 IPL the starter system from disk.

Step 6 Copy the system support modules and production utilities.

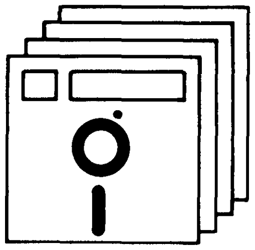
Step 7 Copy the program preparation modules and utilities (this step is required if the Program Preparation Facility 5719-XX6 is being installed).

Step 8 Exercise utilities and program preparation facilities.

Before you begin the installation process, you must have the following diskettes.

Note: If you want to list the contents of the starter diskettes, use the $DISKUT1 utility as described in the *Operator Commands and Utilities Reference.*

5719-XS5 Basic Supervisor and Emulator



Diskette XS5001 contains the supplied starter system and the necessary utilities to install the product.

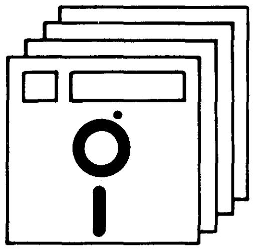
Diskette XS5002 contains supervisor object modules used during the system generation process.

Diskette XS5003 is a multivolume diskette containing two volumes: XS503A and XS503B. Volume XS503A contains object modules that support various system functions. Volume XS503B contains utilities that are included with the supervisor.

Diskette XS5004 contains the remaining utilities included with the supervisor.

Diskette XS5005 contains the stand-alone dump program used for APAR reporting (for machines with 512 K or less storage).

5719-XX6 Program Preparation Facility



Diskette XX6001 contains the EDX program preparation modules.

Diskette XX6002 contains copy modules for inclusion in user application programs.

Diskette XX6003 contains copy modules for inclusion in user application programs.

Diskette XX6004 contains the program preparation utilities and the session manager.

You are ready now to install the Event Driven Executive.

Installing EDX *(continued)*

Using EDX Utilities and Installation Procedures

Throughout the installation procedures, several utility programs are used. In the examples of using these utilities, two types of messages are shown: a loading message and a prompting message.

- You load a program by entering the command **$L $*utility*** where *utility* is the name of the utility being loaded. When the system loads the utility, it displays the following message:

```
LOADING name       nP, HH.MM.SS, LP= nnnn, PART=x
```

  Here, name indicates the utility being loaded. nP indicates that the utility is n number of pages long (256 bytes equals one page). HH.MM.SS equals the time in hours, minutes and seconds. LP= nnnn indicates the load point of the utility. PART=x indicates the partition where the utility is loaded. With the exception of nP, the balance of the message changes depending upon where and when the utility was loaded. Due to these changes, the loading messages shown in the examples are for illustrative purposes only.

- During installation, the utilities prompt you for information. In examples where a response is required, the response is highlighted in red. Enter the responses as shown in the example screens, unless you are instructed to do otherwise.

The installation procedures are presented in step-by-step format. Each step has a brief explanation of what you are supposed to do and an accompanying illustration. The illustration shows switch settings or an example of the contents of your display terminal screen while you are performing the step.

Step 1 - Save Your Existing System

If you are migrating from EDX Version 1 or 2, follow the procedures outlined in Chapter 6, "Migrate to Version 5" on page IS-117 before proceeding to Step 2.

If you want to create a backup copy of all or part of your system and data, do so before proceeding to Step 2. The $MOVEVOL and $COPYUT1 utility descriptions in the *Operator Commands and Utilities Reference* and the *Operation Guide* contain procedures for saving your EDX system and related data.

Installing EDX *(continued)*

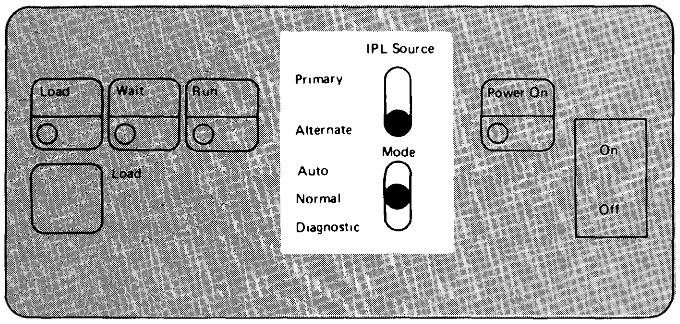
Step 2 - IPL the Starter System

2(a). Set the IPL Source switch and Mode switch to IPL from the diskette unit.

The IPL Source switch has two positions. The primary position selects the primary IPL device for your system. The alternate position selects the alternate IPL device for your system. Depending on how your diskette drive is wired (Primary or Alternate), set the IPL Source switch as appropriate for your system.

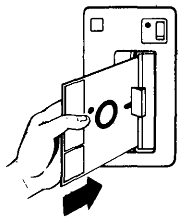
The Mode switch has three positions that allow you to select the mode in which you will operate: Auto IPL, Normal, and Diagnostic. **Set the Mode switch to the Normal position.**

In the example, the diskette unit is the Alternate IPL device. As such, the IPL Source switch is set to Alternate.

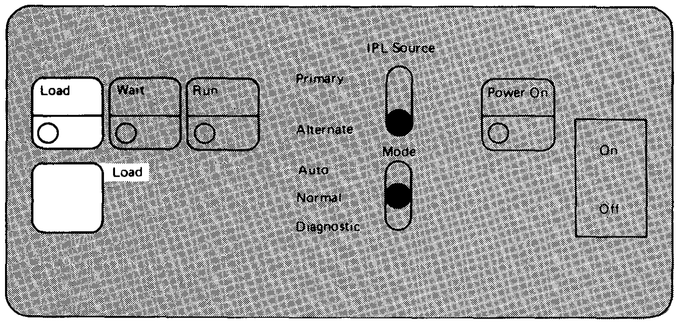


2(b). Insert diskette XS5001 into the diskette unit.

EDX supports the 4964 and 4965 diskette units and the 4966 diskette subsystem. For instructions on inserting and removing diskettes from these devices, refer to the *Operation Guide.*



2(c). Press the **LOAD** button on the Series/1 console.



Installing EDX *(continued)*

When IPL completes, the following system IPL message is displayed on the logging device:

```
*** EVENT DRIVEN EXECUTIVE *** V5.0
***          XPS           ***

IPL = $EDXNUC,XS5001

                XPS SYSTEM STORAGE MAP
                ----------------------
      USER    USER   COMMON  SUPV    SUPV   USER   USER   TOTAL
PART  START   SIZE    SIZE   START   SIZE   START  SIZE   SIZE
  #   (DEC)   (DEC)  (HEX)   (HEX)  (HEX)  (HEX)  (HEX)  (HEX)
  1   47616   17920    0       0     BA00   BA00   4600  10000
  2       0   65536    0       0        0      0  10000  10000
  3       0   65536    0       0        0      0  10000  10000
  4       0   65536    0       0        0      0  10000  10000
TOTAL SIZES (HEX):                   BA00         34600
UNMAPPED STORAGE =     0 (DEC) 2K BLOCKS
EDX INITIALIZATION COMPLETE
```

The IPL message contains 9 columns. An explanation of each column follows.

PART #	Part # indicates the number of the partition.
USER START	The starting address (in decimal) of the first program loaded for execution within each partition.
USER SIZE	The amount of storage (in decimal) within each partition available for program execution.
COMMON SIZE	The size (in hexadecimal) of the common area within each partition.
SUPV START	The starting address (in hexadecimal) of the supervisor within each partition.
SUPV SIZE	The size (in hexadecimal) of the supervisor within each partition.
USER START	The starting address (in hexadecimal) of the first program loaded for execution within each partition.
USER SIZE	The amount of storage (in hexadecimal) within each partition available for program execution.
TOTAL SIZE	The total size (in hexadecimal) of each partition.

Step 3 - Initialize Logical Volumes

Before copying any data sets, you must write a volume directory on disk, allocate volumes, and create directories.

You need to allocate and create directories for the following volumes:

- EDX002
- ASMLIB
- EDX003
- FHVOL (if your disk has fixed heads).

Note: If you are a current user of EDX Version 3, 4.0, or 4.1 you do not need to initialize these volumes; they already exist on your system. However, you must ensure that they are large enough to install EDX Version 5. Use the $DISKUT1 utility (LAV command) to check the recommended volume sizes below and, if they are large enough, skip this step.

If they are not large enough, you will need to reinitialize them. Remember to save any pertinent data before initializing the volumes; once the volumes are reinitialized, the data currently on them is no longer accessible.

Note: Before installing a product, such as the Multiple Terminal Manager or Indexed Access Method, check the amount of space required for each. Refer to the appropriate program directories.

The recommended size (in records) for each volume is as follows:

Volume	Size
EDX002	10,000
ASMLIB	10,000
EDX003	15,640
MACLIB	22,000

Installing EDX *(continued)*

Steps 3(a) through 3(g) show you how to allocate these volumes.

3(a). Load the utility $INITDSK:

1. Press the attention key.
2. Enter **$L $INITDSK**.
3. Press the enter key. (The enter key must be pressed for the Series/1 to read the information you have typed on the screen. For the rest of this procedure and the other procedures in this book, the instruction to "enter" a command includes pressing the enter key.)

```
> $L $INITDSK
LOADING $INITDSK          98P, LP = 9100, PART=1
$INITDSK - DISK INITIALIZATION UTILITY

COMMAND (?):
```

3(b). Initialize the disk.:

1. Enter **ID nn**, where *nn* is the hexadecimal address of the disk you are initializing, as follows:
   - 4962 = 03
   - 4963 or 4967 = 48
   - DDSK-30 or DDSK-60 = 44.

   Note: If you are using a 4962 disk at address X'03' and have a 4963 at address X'48', you must power off the 4963 disk.
2. Verify that you entered the correct disk address and enter a **Y** in response to the verification prompt.
3. Enter a **Y** in response to the DIRECTORY RECORD NUMBER OK? prompt.

$INITDSK writes the volume directory on the disk device and then prompts for allocation of volumes.

```
COMMAND (?):  ID 03

      DEVICE ALREADY INITIALIZED
INITIALIZE DEVICE MAY DESTROY ALL DATA
CONTINUE (Y/N)?  Y

THERE ARE  54000 RECORDS IN YOUR DEVICE

THE DEFAULT OF THE VOLUME DIRECTORY
FOR THIS DEVICE IS RECORD # 129
IT NOW EXISTS AT RECORD # 541
DIRECTORY RECORD NUMBER OK (Y/N)?  Y

DISK INITIALIZED
```

3(c). Allocate volume **EDX002**, initialize the data set directory, allocate a data set for $EDXNUC, and initialize IPL text.

Respond to the prompts as shown.

```
ALLOCATE A VOLUME (Y/N)?  Y
IS THE VOLUME A FIXED HEAD VOLUME (Y/N)?  N
    (Only appears if you have a disk
     with fixed heads)
VOLUME:  EDX002
SIZE IN RECORDS:  10000
EDX002 ALLOCATED
INITIALIZE THE VOLUME JUST ALLOCATED (Y/N)?  Y
MAXIMUM NUMBER OF DATA SETS:  500
DO YOU WANT TO WRITE VERIFY FOR THIS VOLUME (Y/N)?  N
ALLOCATE $EDXNUC (Y/N)?  Y
VOLUME INITIALIZED
INITIALIZE IPL TEXT (Y/N)?  Y
IPL TEXT WRITTEN
```

3(d). Allocate volume **ASMLIB** and initialize the data set directory.

Respond to the prompts as shown.

```
ALLOCATE ANOTHER VOLUME (Y/N)?  Y
IS THE VOLUME A FIXED HEAD VOLUME (Y/N)?  N
    (Only appears if you have a disk
     with fixed heads)
VOLUME:  ASMLIB
SIZE IN RECORDS:  10000
ASMLIB ALLOCATED
INITIALIZE THE VOLUME JUST ALLOCATED (Y/N)?  Y
MAXIMUM NUMBER OF DATA SETS:  500
DO YOU WANT TO WRITE VERIFY FOR THIS VOLUME (Y/N)?  N
ALLOCATE $EDXNUC (Y/N)?  N
VOLUME INITIALIZED
```

3(e). Allocate volume **EDX003** and initialize the data set directory.

Respond to the prompts as shown.

```
ALLOCATE ANOTHER VOLUME (Y/N)?  Y
IS THE VOLUME A FIXED HEAD VOLUME (Y/N)?  N
    (Only appears if you have a disk
     with fixed heads)
VOLUME:  EDX003
SIZE IN RECORDS:  15640
EDX003 ALLOCATED
INITIALIZE THE VOLUME JUST ALLOCATED (Y/N)?  Y
MAXIMUM NUMBER OF DATA SETS:  500
DO YOU WANT TO WRITE VERIFY FOR THIS VOLUME (Y/N)?  N
ALLOCATE $EDXNUC (Y/N)?  N
VOLUME INITIALIZED
```

Installing EDX *(continued)*

3(f). Does your disk have fixed heads?

- If your disk does not have fixed heads, enter an **N** in response to the ALLOCATE ANOTHER VOLUME? prompt, end the $INITDSK utility as shown, and proceed to Step 4.

- If your disk has fixed heads (4962-1F, 4962-2F, 4963-23, or 4963-58), enter a **Y** in response to the ALLOCATE ANOTHER VOLUME? prompt and continue with step 3(g).

```
ALLOCATE ANOTHER VOLUME (Y/N)?  N

COMMAND (?):  EN

$INITDSK ENDED
```

3(g). Allocate volume **FHVOL**. Respond to the prompts as shown.

```
IS THE VOLUME A FIXED HEAD VOLUME (Y/N)?  Y
    (Only appears if you have a disk
     with fixed heads)
VOLUME:  FHVOL
FHVOL ALLOCATED
INITIALIZE THE VOLUME JUST ALLOCATED (Y/N)?  Y
MAXIMUM NUMBER OF DATA SETS:  50
DO YOU WANT TO WRITE VERIFY FOR THIS VOLUME (Y/N)?  N
ALLOCATE $EDXNUC (Y/N)?  N
VOLUME INITIALIZED
ALLOCATE ANOTHER VOLUME (Y/N)?  N

COMMAND (?):  EN

$INITDSK ENDED
```

Step 4 - Copy the Starter Supervisor, Basic Utilities, and Support Modules

Use the $COPYUT1 utility to copy the starter system $EDXNUC on XS5001 to $EDXNUC on EDX002, along with the basic utilities and support modules.

4(a). Load the utility $COPYUT1:

1. Press the attention key.
2. Enter **$L $COPYUT1**.
3. Respond to the $COPYUT1 prompts as shown.

```
> $L $COPYUT1
LOADING $COPYUT1        59P, LP= 9100, PART=1
$COPYUT1 - DATA SET COPY UTILITY

           ** WARNING **
MEMBERS ON TARGET VOLUME WILL BE DELETED
REALLOCATION AND COPYING OF MEMBERS IS
DEPENDENT ON SUFFICIENT CONTIGUOUS SPACE

THE DEFINED SOURCE VOLUME IS XS5001, OK (Y/N)?  Y
THE DEFINED TARGET VOLUME IS XS5001, OK (Y/N)?  N EDX002
MEMBER WILL BE COPIED FROM XS5001 TO EDX002, OK (Y/N)?  Y

COMMAND (?):  ROLLON
```

4(b). Copy $EDXNUC to EDX002 Enter the copy command as shown.

```
COMMAND (?):  CM $EDXNUC *
COPY COMPLETE            400 RECORDS COPIED
```

4(c). Copy the basic utilities and support modules. Respond as shown.

```
COMMAND (?):  CALL

COPY FROM BEGINNING (Y/N)?  Y
```

$COPYUT1 displays a list of the modules being copied from XS5001 to EDX002.

After all the modules on XS5001 are copied to EDX002, $COPYUT1 displays the COMMAND (?): prompt.

Install EDX

Installing EDX *(continued)*

4(d). End the $COPYUT1 utility as shown.

```
COMMAND (?):  EN

$COPYUT1 ENDED
```

- If you are a current user of Version 3, 4.0, or 4.1 and did not perform "Step 3 - Initialize Logical Volumes" on page IS-21, continue with Step 4(e) to initialize the IPL text using the II command of the $INITDSK utility.

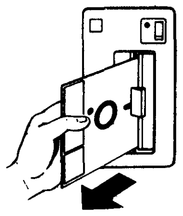
- If you did perform "Step 3 - Initialize Logical Volumes" on page IS-21, skip to Step 4(f).

4(e). Load the utility **$INITDSK**:

1. Press the attention key.
2. Enter **$L $INITDSK**.
3. Respond to the $INITDSK prompts as shown.

```
> $L $INITDSK
LOADING $INITDSK    98P, LP= 9200, PART= 1

$INITDSK - DISK INITIALIZATION UTILITY

COMMAND (?):  II
NUCLEUS:  $EDXNUC
VOLUME:  EDX002
IPL TEXT WRITTEN

COMMAND (?):  EN
$INITDSK ENDED AT 01:05:16
```

4(f). Remove diskette XS5001 from the diskette unit.

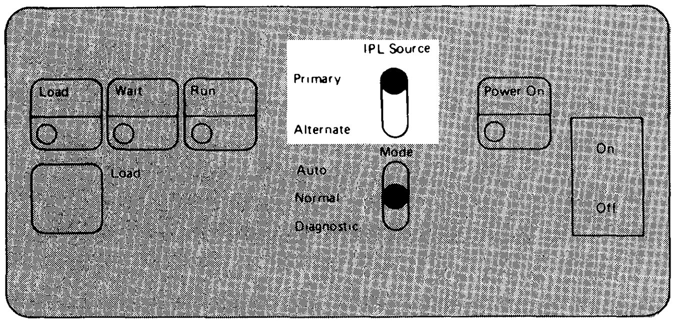


Installing EDX *(continued)*

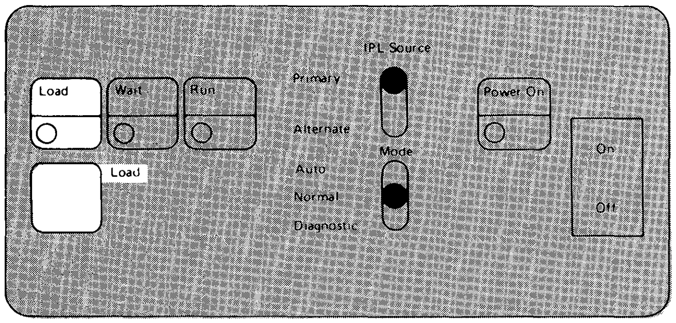
Step 5 - IPL the Starter Supervisor from Disk

5(a). Set the IPL Source switch to IPL from the disk unit.

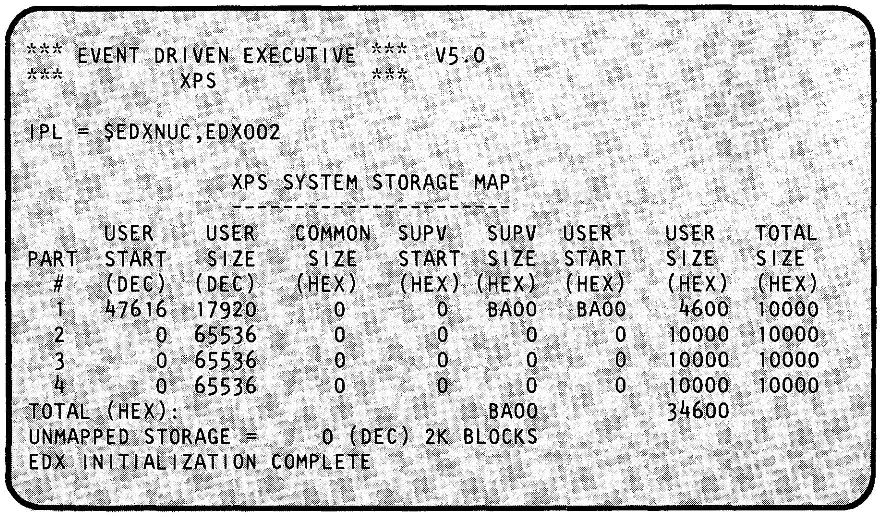
In this example, the disk is wired as the Primary IPL device. Therefore, the IPL Source switch is set to Primary.



5(b). Press the **LOAD** button on the Series/1 console.



When IPL completes, the system issues the illustrated messages indicating that the starter system is successfully installed.



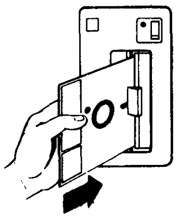
```
*** EVENT DRIVEN EXECUTIVE ***   V5.0
***          XPS           ***

IPL = $EDXNUC,EDX002

                  XPS SYSTEM STORAGE MAP
                  ----------------------
      USER   USER   COMMON  SUPV   SUPV  USER    USER   TOTAL
PART  START  SIZE    SIZE   START  SIZE  START   SIZE   SIZE
  #   (DEC)  (DEC)   (HEX)  (HEX) (HEX)  (HEX)   (HEX)  (HEX)
  1   47616  17920     0       0   BA00   BA00    4600  10000
  2       0  65536     0       0      0      0   10000  10000
  3       0  65536     0       0      0      0   10000  10000
  4       0  65536     0       0      0      0   10000  10000
TOTAL (HEX):                       BA00          34600
UNMAPPED STORAGE =     0 (DEC) 2K BLOCKS
EDX INITIALIZATION COMPLETE
```

Installing EDX *(continued)*

Step 6 - Copy the System Support Modules and Production Utilities

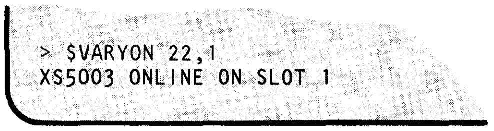
Use the $COPYUT1 utility to copy the system support modules and the production utilities from XS5003 and XS5004.

6(a). Insert diskette **XS5003** into the diskette unit. Diskette XS5003 contains two volumes: XS503A and XS503B. The procedure for copying XS503A to ASMLIB and XS503B to EDX002 follows in Steps 6(b) through 6(c).



Note: If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. To do this:

1. Press the attention key.
2. Use the $VARYON command to vary the diskette on line.
3. Press the enter key.



```
> $VARYON 22,1
XS5003 ONLINE ON SLOT 1
```

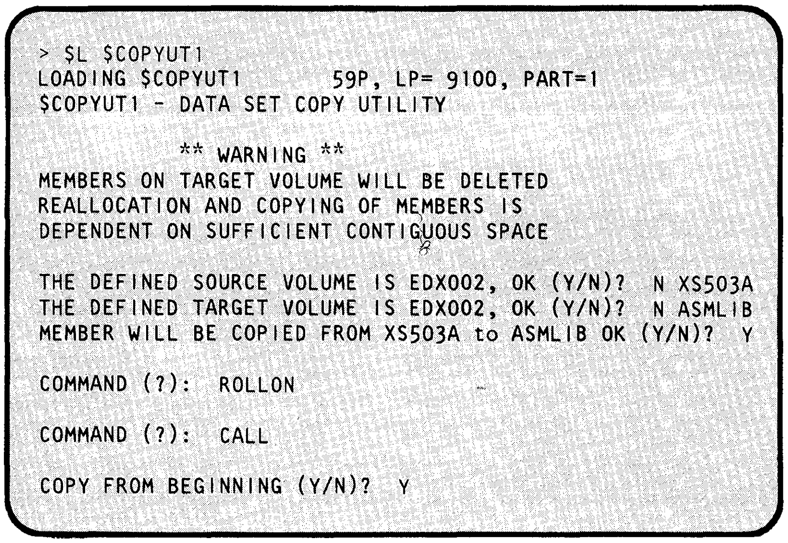
6(b). Load the utility $COPYUT1 and copy the contents of XS503A.

1. Press the attention key.

2. Enter **$L $COPYUT1**.

3. Respond to the $COPYUT1 prompts as shown.

$COPYUT1 displays a list of the modules being copied from XS503A to ASMLIB. After all the modules on XS503A are copied to ASMLIB $COPYUT1 displays the COMMAND (?): prompt.



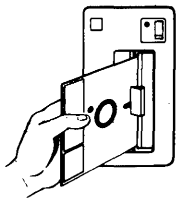
```
> $L $COPYUT1
LOADING $COPYUT1      59P, LP= 9100, PART=1
$COPYUT1 - DATA SET COPY UTILITY

             ** WARNING **
MEMBERS ON TARGET VOLUME WILL BE DELETED
REALLOCATION AND COPYING OF MEMBERS IS
DEPENDENT ON SUFFICIENT CONTIGUOUS SPACE

THE DEFINED SOURCE VOLUME IS EDX002, OK (Y/N)?  N XS503A
THE DEFINED TARGET VOLUME IS EDX002, OK (Y/N)?  N ASMLIB
MEMBER WILL BE COPIED FROM XS503A to ASMLIB OK (Y/N)?  Y

COMMAND (?):  ROLLON

COMMAND (?):  CALL

COPY FROM BEGINNING (Y/N)?  Y
```

6(c). Change the volume and copy the contents of **XS503B**. Respond to the $COPYUT1 prompts as shown.

$COPYUT1 displays a list of the modules being copied from XS503B to EDX002. After all the modules on XS503B are copied to EDX002, $COPYUT1 displays the COMMAND (?): prompt.

```
COMMAND (?):  CV

THE DEFINED SOURCE VOLUME IS XS503A, OK (Y/N)?  N XS503B
THE DEFINED TARGET VOLUME IS ASMLIB, OK (Y/N)?  N EDX002
MEMBER WILL BE COPIED FROM XS503B to EDX002 OK (Y/N)?  Y

COMMAND (?):  CALL

COPY FROM BEGINNING (Y/N)?  Y
```

6(d). Replace diskette **XS5003** with diskette **XS5004** in the diskette unit.



Note: If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. To do this:

1. Press the attention key.
2. Press the enter key.
3. Use the $VARYON command to vary the diskette online.

```
COMMAND (?):
 > $VARYON 22,1
XS5004 ONLINE ON SLOT 1
```

6(e). Change the volume and copy the contents of **XS5004**. Respond to the $COPYUT1 prompts as shown.

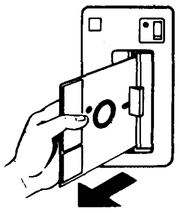
$COPYUT1 displays a list of the modules being copied from XS5004 to EDX002. After the modules from XS5004 are copied to EDX002, $COPYUT1 displays the COMMAND (?): prompt.

```
COMMAND (?):  CV

THE DEFINED SOURCE VOLUME IS XS503B, OK (Y/N)?  N XS5004
THE DEFINED TARGET VOLUME IS EDX002, OK (Y/N)?  Y
MEMBER WILL BE COPIED FROM XS5004 TO EDX002 OK (Y/N)?  Y

COMMAND (?):  CALL

COPY FROM BEGINNING (Y/N)?  Y
```

Installing EDX *(continued)*

6(f) Remove diskette **XS5004** from the diskette unit.

- If you are going to develop application programs on your Series/1 or create a tailored operating system, you need to copy the Program Preparation modules. Proceed to Step 7.
- If not, you can end $COPYUT1 as shown.



```
COMMAND (?):  EN

$COPYUT1 ENDED
```

Step 7 - Copy the Program Preparation Modules and Utilities (5719-XX6)

This step is required only if you are installing 5719-XX6.

Now you can copy the program preparation modules and utilities. These modules are required if you are going to develop application programs on your Series/1.

Note: Either the Program Preparation Facilities (5719-XX6) or the Series/1 Macro Assembler (5719-ASA) is required if you are going to create a tailored operating system or prepare application programs. To install the Event Driven Executive Macro Assembler, follow the directions in the *5719-ASA Program Directory.*

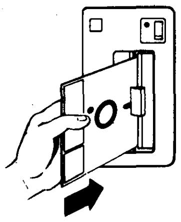
To copy the program preparation modules from XX6001, XX6002, and XX6003 to ASMLIB, and the program preparation utilities, the session manager, and XX6004 to EDX002, perform the following steps:

7(a). Insert diskette **XX6001** into the diskette unit.

Note: If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. To do this:

1. Press the attention key.
2. Press the enter key.
3. Use the $VARYON command to vary the diskette online.

03070



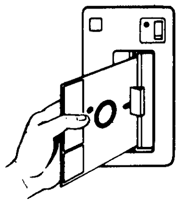
```
COMMAND (?):
> $VARYON 22,1
XX6001 ONLINE ON SLOT 1
```

7(b). Change volumes and copy the contents of **XX6001**. Respond to the $COPYUT1 prompts as shown.

$COPYUT1 displays a list of the $EDXASM modules being copied from XX6001 to ASMLIB. After all the modules are copied from XX6001, $COPYUT1 displays the COMMAND (?): prompt.

```
COMMAND (?):  CV

THE DEFINED SOURCE VOLUME IS XS5004, OK (Y/N)?  N XX6001
THE DEFINED TARGET VOLUME IS EDX002, OK (Y/N)?  N ASMLIB
MEMBER WILL BE COPIED FROM XX6001 to ASMLIB OK (Y/N)?  Y

COMMAND (?):  CALL

COPY FROM BEGINNING (Y/N)?  Y
```

7(c). Replace diskette **XX6001** with diskette **XX6002** in the diskette unit.



Note: If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. To do this:

1. Press the attention key.
2. Press the enter key.
3. Use the $VARYON command to vary the diskette online.

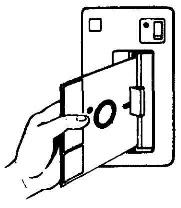
```
COMMAND (?):
 > $VARYON 22,1
XX6002 ONLINE ON SLOT 1
```

7(d). Change the volume and copy the contents of **XX6002**. Respond to the $COPYUT1 prompts as shown. $COPYUT1 displays a list of the modules being copied from XX6002 to ASMLIB. After all the modules are copied from XX6002 to ASMLIB, $COPYUT1 displays the COMMAND (?): prompt.

```
COMMAND (?):  CV

THE DEFINED SOURCE VOLUME IS XX6001, OK (Y/N)?  N XX6002
THE DEFINED TARGET VOLUME IS ASMLIB, OK (Y/N)?  Y
MEMBER WILL BE COPIED FROM XX6002 to ASMLIB OK (Y/N)?  Y

COMMAND (?):  CALL

COPY FROM BEGINNING (Y/N)?  Y
```

Installing EDX *(continued)*

7(e). Replace diskette **XX6002** with diskette **XX6003** in the diskette unit.



Note: If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. To do this:

1. Press the attention key.
2. Press the enter key.
3. Use the $VARYON command to vary the diskette online.

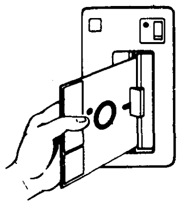
```
COMMAND (?):
> $VARYON 22,1
XX6003 ONLINE ON SLOT 1
```

7(f). Change the volume and copy the contents of **XX6003**. Respond to the $COPYUT1 prompts as shown.

$COPYUT1 displays a list of the modules being copied from XX6003 to ASMLIB. These modules are the copy code modules required for certain applications. After all the modules are copied from XX6003 to ASMLIB, $COPYUT1 displays the COMMAND (?): prompt.

```
COMMAND (?):  CV

THE DEFINED SOURCE VOLUME IS XX6002, OK (Y/N)?  N XX6003
THE DEFINED TARGET VOLUME IS ASMLIB, OK (Y/N)?  Y
MEMBER WILL BE COPIED FROM XX6003 to ASMLIB OK (Y/N)?  Y

COMMAND (?):  CALL

COPY FROM BEGINNING (Y/N)?  Y
```

7(g). Replace diskette **XX6003** with diskette **XX6004** in the diskette unit.



Note: If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. To do this:

1. Press the attention key.
2. Press the enter key.
3. Use the $VARYON command to vary the diskette online.

```
COMMAND (?):
> $VARYON 22,1
XX6004 ONLINE ON SLOT 1
```

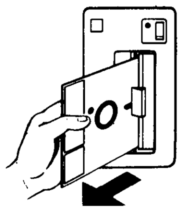
Installing EDX *(continued)*

7(h). Change the volume and copy the contents of **XX6004**. Respond to the $COPYUT1 prompts as shown.

$COPYUT1 displays a list of the modules being copied from XX6004 to EDX002. These modules are the program preparation utilities and the session manager. After all the modules are copied from XX6004 to ASMLIB, $COPYUT1 displays the COMMAND (?): prompt.

```
COMMAND (?):  CV

THE DEFINED SOURCE VOLUME IS XX6003, OK (Y/N)?   N XX6004
THE DEFINED TARGET VOLUME IS ASMLIB, OK (Y/N)?   N EDX002
MEMBER WILL BE COPIED FROM XX6004 to EDX002 OK (Y/N)?  Y

COMMAND (?):  CALL

COPY FROM BEGINNING (Y/N)?  Y
```

7(i). Remove diskette **XX6004** from the diskette unit.



- If you do not wish to install the macro library or the macro assembler, end the $COPYUT1 utility as shown.

- If you are going to install the *Series/1 Macro Assembler*, follow the directions in the 5719-ASA program directory. If you are going to install the *Series/1 Macro Library*, follow the directions in the 5719-LM9 program directory.

```
COMMAND (?):  EN

$COPYUT1 ENDED
```

Step 8 - Exercise the Utilities and Program Preparation Facilities

This optional step exercises the utilities and program preparation facilities and verifies that you copied them correctly from the product diskettes.

To perform this exercise, you need to allocate four data sets and use the verification program CALCDEMO.

Installing EDX *(continued)*

The data sets you need to allocate on EDX002 are:

- EDITWORK - a work data set used by the $FSEDIT utility
- ASMWORK - a work data set used by the $EDXASM compiler
- ASMOBJ - a data set where compiled output from $EDXASM is placed
- LINKWORK - a work data set used by the $EDXLINK linkage editor.

Except for EDITWORK, the data sets are used when you assemble and link edit CALCSRC. CALCSRC is the source for the CALCDEMO program.

Note: By allocating these data sets at this point, you will not need to allocate them later if you generate a tailored operating system or prepare application programs.

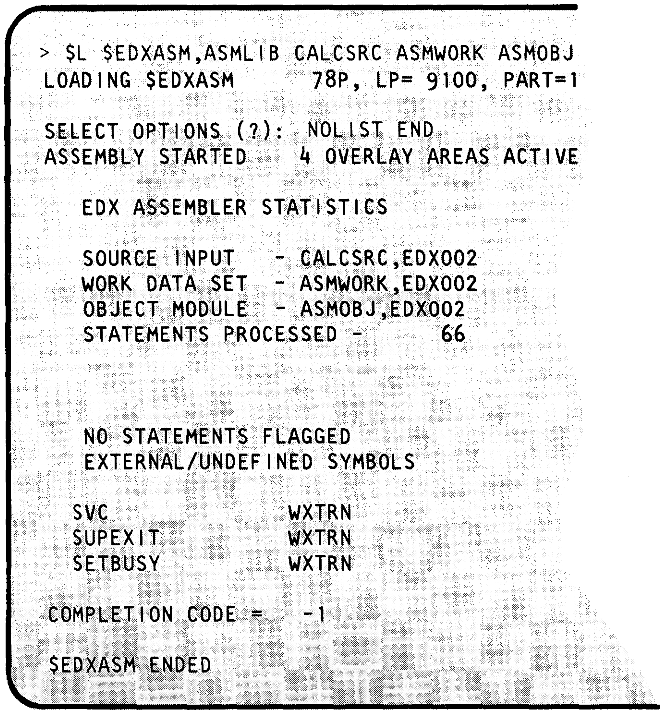
8(a). To allocate the data sets, press the attention key and enter **$L $DISKUT1**. Respond to the $DISKUT1 prompts as shown.

- If you can load $DISKUT1 and allocate the four data sets successfully, continue with Step 8(b).
- If you cannot load $DISKUT1, you may have copied XS5001 to the wrong volume. **Repeat Steps 3 through 7.**

```
> $L $DISKUT1
LOADING $DISKUT1      63P, LP= 9100, PART=1
$DISKUT1 - DATA SET MANAGEMENT UTILITY I

USING VOLUME EDX002

COMMAND (?):  AL EDITWORK 200 D
EDITWORK CREATED

COMMAND (?):  AL ASMWORK 500 D
ASMWORK CREATED

COMMAND (?):  AL ASMOBJ 300 D
ASMOBJ CREATED

COMMAND (?):  AL LINKWORK 600 D
LINKWORK CREATED

COMMAND (?):  EN

$DISKUT1 ENDED
```

8(b). Press the attention key and load the **$EDXASM** compiler using the data sets as shown Respond to the prompts as shown to compile CALCSRC.

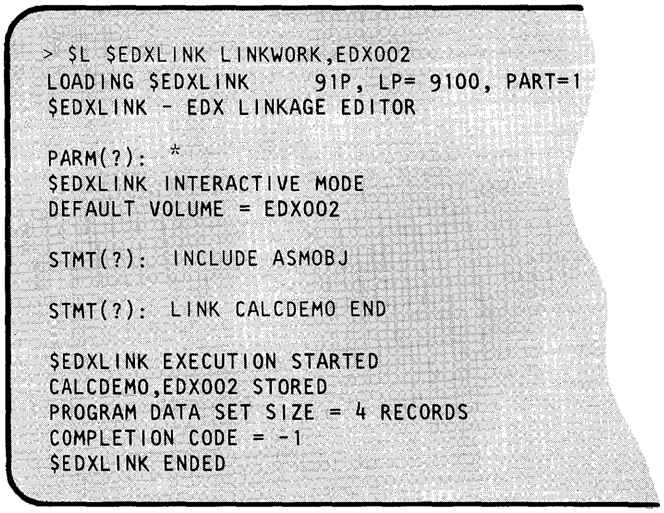
- If you load $EDXASM and compile CALCDEMO successfully (completion code = -1), continue with Step 8(c).
- If you cannot load $EDXASM, you may have copied XX6001 through XX6004 to the wrong volumes. **Repeat Steps 3 through 7.**



```
> $L $EDXASM,ASMLIB CALCSRC ASMWORK ASMOBJ
LOADING $EDXASM      78P, LP= 9100, PART=1

SELECT OPTIONS (?):  NOLIST END
ASSEMBLY STARTED    4 OVERLAY AREAS ACTIVE

   EDX ASSEMBLER STATISTICS

   SOURCE INPUT    - CALCSRC,EDX002
   WORK DATA SET   - ASMWORK,EDX002
   OBJECT MODULE   - ASMOBJ,EDX002
   STATEMENTS PROCESSED -      66


   NO STATEMENTS FLAGGED
   EXTERNAL/UNDEFINED SYMBOLS

  SVC               WXTRN
  SUPEXIT           WXTRN
  SETBUSY           WXTRN

COMPLETION CODE =    -1

$EDXASM ENDED
```

8(c). Press the attention key, load **$EDXLINK** using the data set as shown. Respond to the prompts as shown to link edit CALCSRC. This step produces a listing on the printer designated as $SYSPRTR.

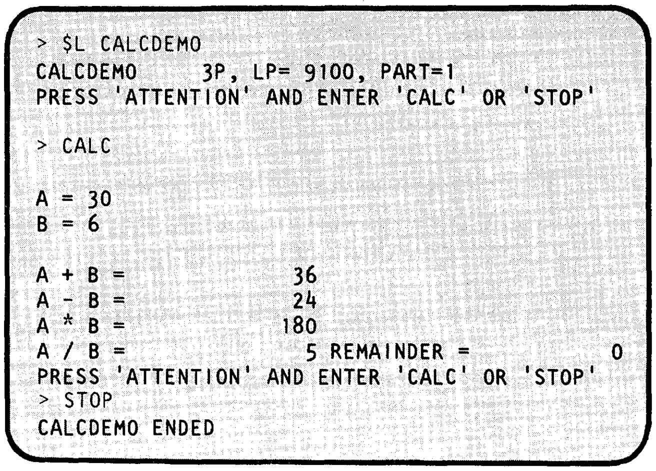
Note: If you have created CALCDEMO previously, respond with LINK CALCDEMO REP END to the second STMT(?): prompt.



```
> $L $EDXLINK LINKWORK,EDX002
LOADING $EDXLINK      91P, LP= 9100, PART=1
$EDXLINK - EDX LINKAGE EDITOR

PARM(?):  *
$EDXLINK INTERACTIVE MODE
DEFAULT VOLUME = EDX002

STMT(?):  INCLUDE ASMOBJ

STMT(?):  LINK CALCDEMO END

$EDXLINK EXECUTION STARTED
CALCDEMO,EDX002 STORED
PROGRAM DATA SET SIZE = 4 RECORDS
COMPLETION CODE = -1
$EDXLINK ENDED
```

Installing EDX *(continued)*

8(d). Press the attention key and enter **$L CALCDEMO**. Follow its operating instructions until you decide to end the program. When you decide to end the program, enter **STOP**.



```
> $L CALCDEMO
CALCDEMO      3P, LP= 9100, PART=1
PRESS 'ATTENTION' AND ENTER 'CALC' OR 'STOP'

> CALC

A = 30
B = 6

A + B =              36
A - B =              24
A * B =             180
A / B =               5 REMAINDER =          0
PRESS 'ATTENTION' AND ENTER 'CALC' OR 'STOP'
> STOP
CALCDEMO ENDED
```

Now that EDX is installed, you are ready either to generate a tailored operating system or to prepare application programs for execution on your Series/1. To generate a tailored operating system, see Chapter 4, "Select Your Required Support" on page IS-39. For information on preparing programs, refer to the *Event Driven Executive Language Programming Guide.*

Program Function Keys for the 4979 Display Station

If you use the starter system as your operating system and have a 4979 Display Station at address 04, the program function keys operate as follows:

Key label	Function
PF1	PF1
PF2	PF3
PF3	PF5
PF4	PF2
PF5	PF4
PF6	PF6

Terminal Initialization for the Starter System

All terminals are initialized as part of the IPL process. In the starter system, the primary logging terminal to receive/display all exception messages ($SYSLOG) is defined as a 4978 Display Station at hardware address X'04'. If there is no terminal at hardware address X'04', terminal initialization support relocates $SYSLOG to a 4978 Display Station at hardware address X'24'. If there is no terminal at hardware address X'24', terminal initialization support relocates $SYSLOG to a 4980 Display Station at hardware address X'80'.

If an error occurs when writing to $SYSLOG, terminal initialization reverts to the alternate logging device ($SYSLOGA). In the starter system, $SYSLOGA is defined as a teletypewriter at hardware address X'00'.

Note: If a TTY attachment exists at address 0 without a terminal attached to it, the system cannot detect an error when writing to $SYSLOGA, terminal initialization does not revert to $SYSLOGB, and messages are not displayed on any terminals.

If an error occurs when writing to $SYSLOGA, terminal initialization reverts to the second alternate logging device ($SYSLOGB). In the starter system, $SYSLOGB is defined as a 3101 Display Terminal in block mode at hardware address X'59', connected via a Multifunction Attachment (MFA) at hardware address X'58'. If no MFA exists at hardware address X'58', terminal initialization support relocates $SYSLOGB to a 3101 Display Terminal in block mode, connected to a single line ACCA (#1610) attachment at hardware address X'08'. If no ACCA attachment exists at hardware address X'08', terminal initialization support relocates $SYSLOGB to a 3101 Display Terminal in block mode, connected to a Multiline Terminal Adapter (2091/2092) at hardware address X'60'. If no MLTA attachment exists at hardware address X'60', terminal initialization support relocates $SYSLOGB to a 3101 Display Terminal in block mode connected to a Feature Programmable Communications Adapter (2095/2096) at hardware address X'68'.

Notes

Chapter 4. Select Your Required Support

System Features

This chapter helps you select the system features you need for your tailored operating system. System features include defining processor storage characteristics, the I/O devices attached to your Series/1, the software features you require for your application, and the EDX-related products you are including as part of your tailored operating system.

The Remote Support Link

The Remote Support Link allows an IBM support center representative to dial into your Series/1. Using a switched telephone network, he can issue operator commands, execute EDX programs, and transfer disk data sets and messages. The required TERMINAL definition statements for this link are discussed in "Step 2 - Defining Your System Configuration" on page IS-42; the supervisor modules you must include are discussed in "Translation Table Support" on page IS-56 and "Terminal Support" on page IS-57. A detailed description of the Remote Support Link is presented in the *Problem Determination Guide* and the *IBM Series/1 Software Service Guide*, GC34-0099.

Designing Your Operating System

The steps you perform in designing your operating system are:

Step 1 Estimate your supervisor's storage requirements.

Step 2 Define your system configuration.

Designing Your Operating System *(continued)*

Step 3 Select your software support.

Step 4 Define the structure of your supervisor, if applicable.

Note: Be sure to review the program directories for all EDX-related products you are installing to determine any special considerations before generating a tailored operating system.

To help you design your operating system, the following work sheets are provided in Appendix E:

- Use work sheet 1 to estimate the size of your supervisor.
- Use work sheet 2 to define the characteristics and partition structure of processor storage and the I/O devices attached to your Series/1. Use work sheet 2 with $EDXDEF to tailor your operating system.
- Use work sheet 3 to define the software features you require to support the defined I/O devices and the EDX-related products you include in your system. Use work sheet 3 to select the supervisor modules to be included in $LNKCNTL.
- Use work sheet 4 to estimate the amount of storage required by supervisor object modules that are located outside of partition 1 and the amount of storage required by programs to execute within a partition.

If you decide to locate your supervisor totally in partition 1, skip step 5. Complete work sheets 2 and 3 and follow the procedures in Chapter 5, "Generate a Tailored Operating System" on page IS-77.

However, before you can select your required support, you should be familiar with the characteristics of processor storage.

Processor Storage

Processor storage is divided into partitions. A partition is a contiguous, fixed-length area of storage (maximum of 64K or 65,536 bytes) that is used for execution of programs. You can define up to eight partitions for the 4955, or 4956, up to four partitions for the 4954, and up to two partitions for the 4952.

Note: You can define up to 16 partitions for Extended Address Mode. Refer to the *Extended Address Mode and Performance Analyzer User Guide* for detailed information.

The supervisor (or a portion of the supervisor) is always located in partition 1. The storage available in all defined partitions for program execution is limited to 64K bytes minus the number of bytes occupied by the supervisor. Each partition is defined in multiples of 2K bytes and can contain more than one program simultaneously within the limits of the storage assigned to it.

Sample System

This chapter contains an example of designing a tailored operating system to match specific hardware requirements and software features. When applicable, excerpts from work sheet 2 are shown as examples of how to code a definition statement for a specific I/O device. At the end of the chapter, marked up copies of work sheets 2 and 3 show the definition statements and the software features that define the sample system.

The sample assumes that the hardware and software requirements are as follows:

Hardware Requirements: Our sample Series/1 is a 4956 processor with 512K bytes of storage. The devices supported and their address assignments are:

Device	Address
4962-1F Disk Storage Unit	03
4964 Diskette Unit	02
4978 Display Station	24
4979 Display Station	04
4973 Line Printer	21

The 4979 Display Station is defined as the primary system logging device ($SYSLOG) to receive all exception messages. The 4978 Display Station is defined as the alternate system logging device ($SYSLOGA).

The 4973 Printer is defined as the system printer ($SYSPRTR) to receive the hard-copy output from all system programs.

Software Support Requirements: The following software features are required for the sample system:

Software Features	Requirements
Multipartition supervisor	Supervisor code in partitions 1 and 8
Common area	Partitions 2, 4, and 5
Number of partitions and sizes	5 (0,24,16,20,32)
Number of programs per partition	(1,10,10,5,5)
Unmapped storage	YES

Each disk device will have its own task for input/output.

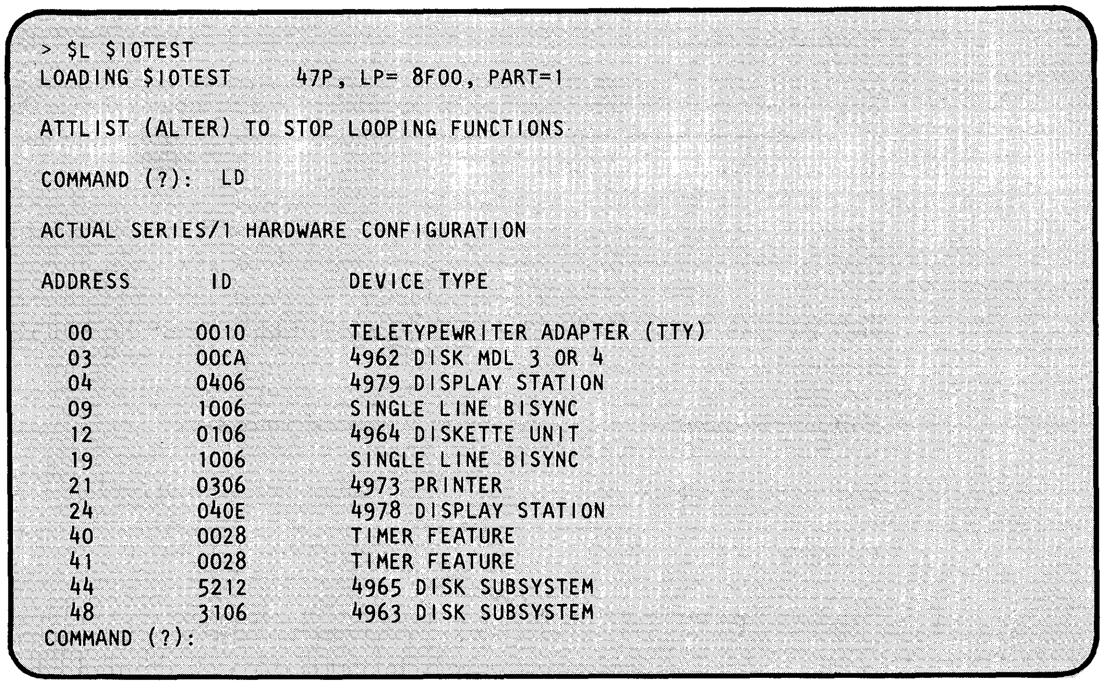
Select Your Required Support

Step 1 - Estimating Total Supervisor Size

Your first step is to estimate the total size of your supervisor. Use work sheet 1 which contains a list of the I/O devices you can attach to the Series/1. Along with each device type is the amount of storage the supervisor requires to support that device. An estimate of 64K (65536 bytes) or greater indicates that you should reduce the size of your supervisor. The ways to reduce the size of your supervisor are discussed under "Step 4 - Defining Supervisor Structure" on page IS-68. However, before performing that step, you should perform steps 2 and 3 first in order to be aware of the I/O devices and software support you are including in your operating system.

Step 2 - Defining Your System Configuration

The second step in selecting your required support is to describe the devices attached to your Series/1. To do this, you must know the configuration of the system on which you intend to run the tailored operating system. When coding the system definition statements (work sheet 2), one of the operands you must specify is the device hardware address. Use the $IOTEST utility (LD command) to find out which devices are installed on your system and their addresses. Before $IOTEST can list all the devices attached to your Series/1, the devices must be switched on. The following example shows the devices and their address assignments for the sample system.

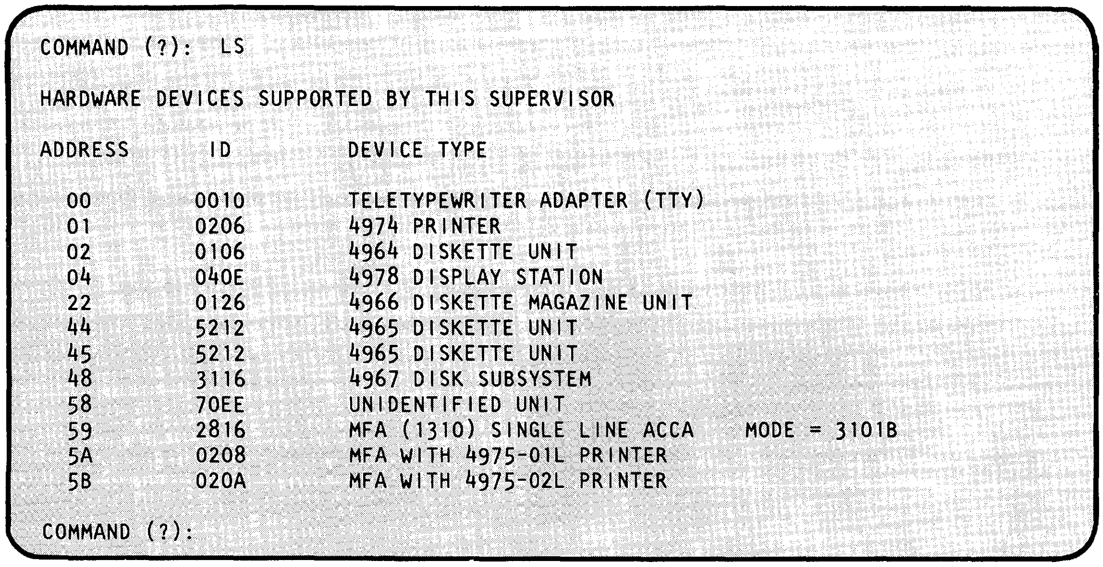


```
> $L $IOTEST
LOADING $IOTEST      47P, LP= 8F00, PART=1

ATTLIST (ALTER) TO STOP LOOPING FUNCTIONS

COMMAND (?):  LD

ACTUAL SERIES/1 HARDWARE CONFIGURATION

ADDRESS       ID          DEVICE TYPE

  00         0010         TELETYPEWRITER ADAPTER (TTY)
  03         00CA         4962 DISK MDL 3 OR 4
  04         0406         4979 DISPLAY STATION
  09         1006         SINGLE LINE BISYNC
  12         0106         4964 DISKETTE UNIT
  19         1006         SINGLE LINE BISYNC
  21         0306         4973 PRINTER
  24         040E         4978 DISPLAY STATION
  40         0028         TIMER FEATURE
  41         0028         TIMER FEATURE
  44         5212         4965 DISK SUBSYSTEM
  48         3106         4963 DISK SUBSYSTEM
COMMAND (?):
```

Step 2 - Defining Your System Configuration *(continued)*

Figure 5 shows an example of using the $IOTEST LS command to list the hardware devices supported by the IBM-supplied supervisor under which $IOTEST is running. Figure 5 shows the devices supported by the starter system $EDXNUC.

By comparing the previous figure and Figure 5 , you can see that the starter system does not support some of the devices in our sample system. If your actual Series/1 configuration is different than the hardware devices supported by the starter system and you want to support these additional devices, you need to define the configuration of your system. Use work sheet 2 to specify the configuration statements for your system.



```
COMMAND (?):  LS

HARDWARE DEVICES SUPPORTED BY THIS SUPERVISOR

ADDRESS      ID        DEVICE TYPE

  00        0010       TELETYPEWRITER ADAPTER (TTY)
  01        0206       4974 PRINTER
  02        0106       4964 DISKETTE UNIT
  04        040E       4978 DISPLAY STATION
  22        0126       4966 DISKETTE MAGAZINE UNIT
  44        5212       4965 DISKETTE UNIT
  45        5212       4965 DISKETTE UNIT
  48        3116       4967 DISK SUBSYSTEM
  58        70EE       UNIDENTIFIED UNIT
  59        2816       MFA (1310) SINGLE LINE ACCA    MODE = 3101B
  5A        0208       MFA WITH 4975-01L PRINTER
  5B        020A       MFA WITH 4975-02L PRINTER

COMMAND (?):
```

Figure 5. Hardware Devices Supported by the Starter System ($EDXNUC)

System Definition Statements

The following definition statements are used to describe processor storage and I/O devices to your operating system:

- ADAPTER - Defines a multiline attachment
- BSCLINE - Defines a binary synchronous line
- DISK - Defines direct access storage devices
- EXIODEV - Defines EXIO interface devices
- HOSTCOMM - Defines host communication support
- SENSORIO - Defines sensor I/O devices
- SYSTEM - Defines processor characteristics
- TAPE - Defines tape devices
- TERMINAL - Defines terminals
- TIMER - Defines system timer feature.

Step 2 - Defining Your System Configuration *(continued)*

A brief description of each of these definition statements is presented in this chapter. A detailed description of the statements, along with the syntax and operand definitions, is provided in Appendix A.

The definition statements presented here are in the order of appearance in work sheet 2.

SYSTEM Statement

The SYSTEM statement defines the partition structure, the storage assigned to each partition for execution of application programs, and the number of programs that can execute concurrently in each partition. This statement must be specified only once.

The MAXPROG= operand specifies the maximum number of concurrently executing programs to be allowed in each partition.

The PARTS= operand specifies the number of 2K (2K=2048 bytes) blocks of storage assigned to each partition. The PARTS= operand also defines the partition structure of your system. The supervisor (or a portion of the supervisor) always resides in partition 1. Partition 1 must be large enough to contain the supervisor. If you have only one partition, this partition must be large enough to contain the entire supervisor and your largest application program. If you have multiple partitions, they must be defined in multiples of 2K bytes of storage and can contain more than one program simultaneously within the limits of the storage assigned to each partition.

See "SYSTEM - Define Processor Storage" on page IS-162 for a detailed description of this statement.

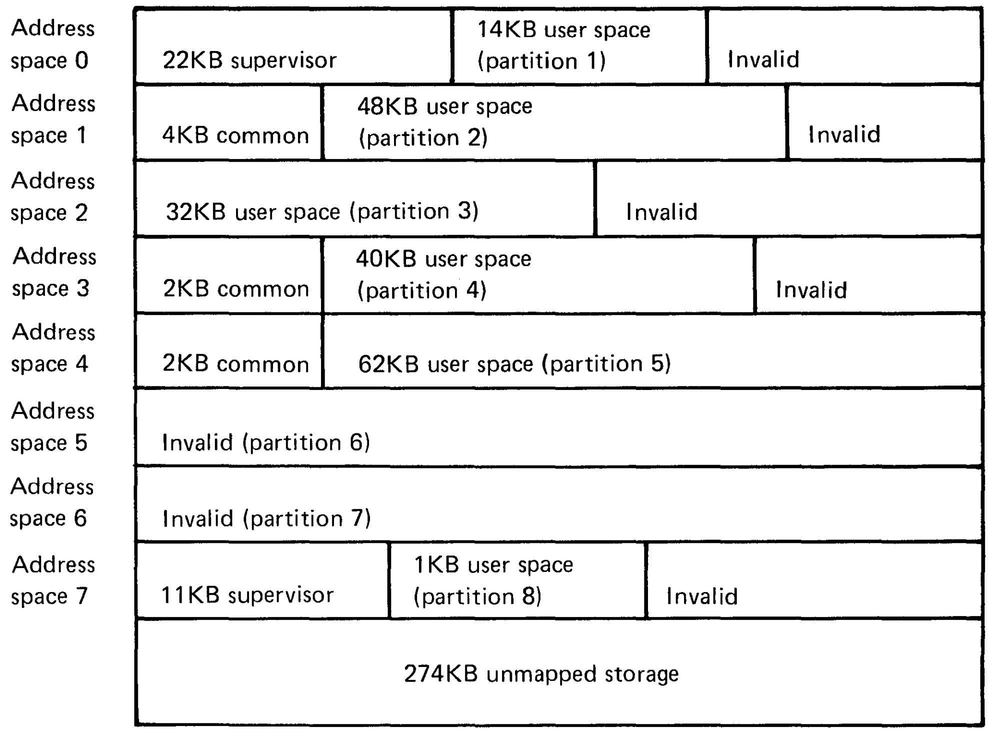
If you determined in "Step 1 - Estimating Total Supervisor Size" on page IS-42 that your supervisor is 64K or greater or that you want to free up storage in partition 1, you may need to modify the PARTS= and MAXPROG= operands of the SYSTEM statement. After you perform "Step 4 - Defining Supervisor Structure" on page IS-68, then you can decide how to structure your processor storage (number of partitions and their sizes) and the number of programs that can execute in each partition.

The sample system is defined as having five partitions (0, 24, 16, 20, 32). For each partition, the maximum number of programs that can execute concurrently are 1, 10, 10, 5, 5, respectively. The SYSTEM statement defined for the sample system is:

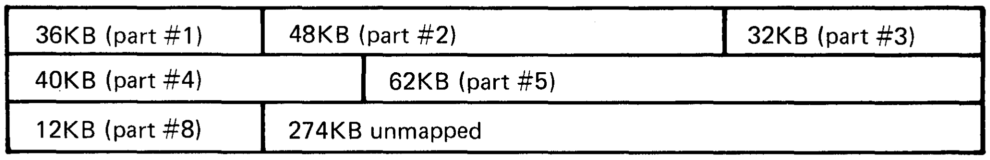
```
         SYSTEM    MAXPROG=(1,10,10,5,5),PARTS=(0,24,16,20,32),    C
               COMMON=(0,2,0,1,1)
```

The sample system logically and physically maps in storage as follows:

Logical mapping:



Physical mapping:



Step 2 - Defining Your System Configuration *(continued)*

1. Partition 1 is defined to contain supervisor code only (PART=0). The supervisor required 11 blocks (22KB) of storage and the initialization routines required 7 blocks (14KB) of storage at IPL time. After initialization, the 7 blocks of storage required by the initialization routines were given back as user space.

2. Partition 2 required 2 blocks (4KB) to serve as addressing pointers to the COMMON area and has 24 blocks (48KB) of actual storage mapped for user space. The 2 blocks of pointers cannot map actual storage since they are reserved as COMMON area pointers. Partition 2 can execute up to 10 programs concurrently.

3. Partition 3 requires 16 blocks (32KB) of storage for user space and can execute up to 10 programs concurrently.

4. Partition 4 requires 2 blocks (4KB) to serve as addressing pointers to the COMMON area and has 20 blocks (40KB) of actual storage mapped for user space. The 2 blocks of pointers cannot map actual storage since they are reserved as COMMON area pointers. Partition 4 can execute up to 5 programs concurrently.

5. Partition 5 requires 2 blocks (4KB) to serve as addressing pointers to the COMMON area and has 31 blocks (62KB) of actual storage mapped for user space. The 2 blocks of pointers cannot map actual storage since they are reserved as COMMON area pointers. Partition 5 can execute up to 5 programs concurrently.

6. Partition 6 is not defined.

7. Partition 7 is not defined.

8. Partition 8 is not defined on the SYSTEM statement. However, a multipartition supervisor was defined with the PART statement in the $LNKCNTL data set as having supervisor code in partition 8. As a result, partition 8 was defined automatically and 11KB of storage was assigned to the supervisor. In addition, because storage is assigned in 2K blocks, 1KB of storage was assigned as user space for a total of 12KB in partition 8.

9. The size of processor storage is 512KB. Only 238KB of storage was defined leaving 274KB of unmapped storage.

TAPE Statement

The TAPE statement defines the 4968 and 4969 tape units. One TAPE statement is required for each tape device attached to your Series/1. Group all TAPE and DISK statements together. The last TAPE or DISK statement must specify END=YES (as shown in work sheet 2).

See "TAPE - Define Tape Device" on page IS-181 for a detailed description of this statement.

DISK Statement

The DISK statement defines disk and diskette devices to the system. One DISK statement is required for each disk or diskette device attached to your Series/1.

The sample system is defined as having a 4962-1F disk and a 4964 diskette unit. These devices are defined on work sheet 2 as follows:

```
          DISK      DEVICE=4962-1F,ADDRESS=03,TASK=YES
          DISK      DEVICE=4964,ADDRESS=02,TASK=YES,END=YES
```

See "DISK - Define Direct Access Storage" on page IS-154 for a detailed description of this statement.

Disk Considerations: You should specify all disks before you specify diskettes in order to get better performance. You can designate a disk volume as a ***performance volume*** by specifying the VOLNAME operand in the DISK definition statement. As with all volumes, you must allocate and initialize this volume using the $INITDSK utility. However, a performance volume is opened at IPL time making the opening of data sets in a performance volume faster.

Each performance volume requires 46 bytes in the supervisor; this additional storage cannot be reclaimed until you generate a new supervisor.

Note: Deleting a performance volume does not allow the storage to be reused.

TIMER Statement

The TIMER statement defines the #7840 Timer Feature as the system timer in the generated system. This statement is used only for defining the #7840 timer. If the system has a native timer (4952, 4954 and 4956 processors) that is used instead of the #7840 timer feature card, you do not need to code this statement.

See "TIMER - Define System Timer Features" on page IS-242 for a detailed description of this statement.

ADAPTER Statement

The ADAPTER statement defines one of the following multiline attachment features:

- Multifunction Attachment Feature #1310
- Printer Attachment - 5200 Series Feature #5640
- Multidrop Work Station Attachment Feature #1250.

Step 2 - Defining Your System Configuration *(continued)*

One ADAPTER statement is required for each adapter attachment attached to your Series/1. All ADAPTER statements must be grouped together and must precede the definition of the first device attached to a specific attachment. The last ADAPTER statement specified must include END=YES.

See "ADAPTER - Define a Multiline Attachment" on page IS-146 for a detailed description of this statement.

TERMINAL Statement

The TERMINAL statement defines each EDX terminal to be supported by the operating system. The DEVICE operand of the TERMINAL statement identifies the type of terminal. EDX supports many different terminals; see "TERMINAL - Define Input/Output Terminals" on page IS-183 for a list of the supported terminals. The required operands and defaults of the TERMINAL statement can be different for each device; therefore, the TERMINAL statement is presented by device type. (See "TERMINAL - Define Input/Output Terminals" on page IS-183 for a detailed description of this statement.)

Notes:

1. The 3101 Display Terminal is connected to the Series/1 by several EDX support attachment features. For each attachment and type of interface, it is necessary to set the 3101 switches, have the attachment card physically jumpered, connect the cables, and specify the appropriate TERMINAL statement. Before specifying a TERMINAL statement for the 3101, see Appendix C, "3101 Configuration Information" on page IS-249.

2. See "ACCA-Type Terminals" on page IS-214 for information about the TERMINAL statement for the Remote Support Link.

Our sample system is defined as having the following terminals:

- a 4973 printer
- a 4979 display station
- a 4978 display station.

The TERMINAL statements that define these I/O devices to our operating system are:

```
$SYSPRTR TERMINAL DEVICE=4973,ADDRESS=21
$SYSLOG  TERMINAL DEVICE=4979,ADDRESS=04,HDCOPY=$SYSPRTR
$SYSLOGA TERMINAL DEVICE=4978,ADDRESS=24,HDCOPY=$SYSPRTR,END=YES
```

BSCLINE Statement

The BSCLINE statement defines a binary synchronous line. One BSCLINE is required for each line to be referenced by programs using the Binary Synchronous Communications Access Method (BSCAM).

See "BSCLINE - Define a Binary Synchronous Line" on page IS-150 for a detailed description of this statement.

EXIODEV Statement

The EXIODEV statement defines the devices to be supported through the EXIO interface. All EXIODEV statements must be grouped together. The last EXIODEV statement must include an END=YES specification.

An EXIODEV statement must be defined for each System/370 Channel Device attached to your system. For more information on defining channel attach devices, refer to the Program Directory for the Series/1-System/370 Channel Attach (5719-CX1).

An EXIODEV statement must be defined for each physical unit that defines the SDLC line. For more information on defining an SDLC line, refer to the Program Directory for the Series/1 Systems Network Architecture, (5719-SX1).

See "EXIODEV - Define EXIO Interface Device" on page IS-157 for a detailed description of this statement.

HOSTCOMM Statement

The HOSTCOMM statement defines the device type and address of the device to be used for host communication support. This support only operates with the Host Communications Facility Installed User Program (IUP).

The Host Communications Facility (HCF) allows an EDL program to communicate with the Host Communication Facility IUP installed on a host IBM S/370. The HCF is used to perform file transfers to and from the host and to submit a job stream to the host. You must use a point-to-point nonswitched BSC line to connect the Series/1 and S/370.

See "HOSTCOMM - Define Host Communications Support" on page IS-159 for a detailed description of this statement.

SENSORIO Statement

The SENSORIO statement defines the sensor I/O devices to be supported. All SENSORIO statements must be grouped together and the last one must include an END=YES specification.

See "SENSORIO - Define Sensor I/O Devices" on page IS-160 for a detailed description of this statement.

Step 2 - Defining Your System Configuration *(continued)*

System Common Data Area

The system common data area is known by the system global name $SYSCOM. $SYSCOM is a data area in an operating system which can be accessed from applications written in EDL, assembler language, COBOL, FORTRAN, Pascal, and PL/I. It is used for communication and synchronization between programs. If you select this option, you can map the portion of the supervisor containing $SYSCOM in partition 1 to the partitions of your choice. See "SYSTEM - Define Processor Storage" on page IS-162 for a description of coding the system common data area.

FORTRAN and assembler language programs can reference the system common data area directly by referencing global section definitions. PL/I and EDL programs must reference the common area indirectly by using the contents of $SYSCOM as a base address and referencing data elements as displacements from this base. Refer to the appropriate language user's guide for examples of using $SYSCOM and global sections.

As shown in work sheet 2, there is an example of specifying a 128-word area called $EDXPTCH which can be used as part of $SYSCOM and should be coded as part of your definition statements. In addition, $SYSCOM consists of two queue control blocks (QCBs) and two event control blocks (ECBs). These control blocks can be used as is or you can change them.

Step 3 - Selecting Your Software Support

The third step in selecting your required support is to choose the software features needed to support your configuration and the EDX-related products you are installing as part of your system. Use work sheet 3 to keep track of the support object modules you need.

If you determined in Step 1 that your supervisor is 64K or greater or that you want to free up storage in partition 1, you need to know which support object modules need to be included in your supervisor. You need this information to complete "Step 4 - Defining Supervisor Structure" on page IS-68.

The object modules that provide the software features supported by EDX are explained on the following pages. For the starter system, these names are contained in the $LNKCNTL data set on volume ASMLIB. Work sheet 3 parallels $LNKCNTL which contains the control statements and object module names used by the linkage editor $XPSLINK. $XPSLINK is a linkage editor that is set up to generate a single or multipartition supervisor and is used only during system generation.

Step 3 - Selecting Your Software Support *(continued)*

Control statements are the instructions used by $XPSLINK to convert the assembled object modules into an executable load module. Work sheet 3 contains the following control statements in the order shown below.

OPTION NOVERLAY	An OPTION NOVERLAY statement specifies that the supervisor will not default to overlay structure.
PART	A PART statement defines the partition number where groupings of supervisor object modules are to be located.
VOLUME	A VOLUME statement defines the default volume for $XPSLINK control statements.
OVLAREA	An OVLAREA statement defines an overlay area. The supervisor defaults to overlay structure.
INCLUDE	An INCLUDE statement identifies an object module to be included in the generated supervisor.
LINK	A LINK statement causes $XPSLINK to perform a link using the control statements within the data set and to store the resulting executable load module (the supervisor) in a data set named $EDXNUCx where x is any alphanumeric character.

With the exception of the OPTION NOVERLAY and the PART statements, these control statements are explained under the $EDXLINK utility in the *Operator Commands and Utilities Reference.*

OPTION NOVERLAY Statement

The OPTION NOVERLAY statement is an $XPSLINK statement and is used only during system generation. If this statement is included in the $LNKCNTL data set, the supervisor is built without the overlay structure. Without overlay structure, $XPSLINK automatically includes required initialization routines in your supervisor as resident programs. This means that the required initialization routines are loaded simultaneously in supervisor storage at IPL time. As a result, they increase the amount of storage required by the supervisor in partition 1. For further information, see "Including Initialization Routines in Your Supervisor" on page IS-65.

PART Statement

The PART statement is an $XPSLINK statement and is used only during system generation. A PART statement defines the partition number where supervisor object modules are to be located and is used to create a multipartition supervisor. One PART statement is required for each partition in which you have supervisor object modules.

Step 3 - Selecting Your Software Support *(continued)*

The number on the PART statement preceding a group of modules specifies the partition where the modules are to be located. Some groups of supervisor object modules must remain located in partition 1; others can be located outside of partition 1. In work sheet 3, the modules are grouped and a statement preceding each group indicates if that group of modules can be located outside of partition 1. To locate a group of modules outside partition 1, you must move the modules to the area specified on the work sheet and in the $LNKCNTL data set. Then enter the number of the partition where you want the group located on the PART statement.

All groups of object modules defined to be in a specific partition must be adjacent to one another. For example, you can specify partitions 1, 1, 2, 2, 3, 4, 4 or 1, 2, 2, 4, 4, 3, 5, 5, 6, 6, 7. You cannot specify partitions 1, 3, 2, 1, 3, 4, 3. This means that if you have already defined a group of modules to be in a specific partition and you wish to place another group of modules in that same partition, you must move the second group of modules to follow the PART statement defining that partition. Also, if you define PART statements, you must specify PART 1 first.

Syntax:

```
PART partnumber

Required: partnumber
Default:  none
```

Example: Define specific groups of object modules in partitions 3 and 4.

```
*------------------------------------------------------------
* SUPERVISOR CODE BEING MOVED OUT OF
* PARTITION 1 'MUST' BE MOVED TO HERE
*------------------------------------------------------------
*
*------------------------------------------------------------
* SENSOR INPUT/OUTPUT - MUST BE GROUPED TOGETHER IN ANY PARTITION
*------------------------------------------------------------
PART 3
INCLUDE SBCOM        *15*  BASIC SENSOR I/O SUPPORT
INCLUDE SBAI         *3*   ANALOG INPUT SUPPORT
INCLUDE SBAO         *3*   ANALOG OUTPUT SUPPORT
INCLUDE SBIDO        *3*   DIGITAL INPUT/OUTPUT SUPPORT
INCLUDE SBPI         *3*   PROCESS INTERRUPT SUPPORT
*------------------------------------------------------------
* BISYNC COMMUNICATION _ MAY BE IN ANY PARTITION
*------------------------------------------------------------
PART 4
INCLUDE BSCAM        *7*   BISYNC COMMUNICATION SUPPORT
*------------------------------------------------------------
```

VOLUME Statement

The VOLUME statement defines the default volume for $XPSLINK control statements.

XS5002	XS5002 is the default volume for the object modules. This statement is required and must be included in every supervisor.

OVLAREA Statement

The OVLAREA statement is an optional link-control statement used to define an overlay area within the supervisor. This area is used to process initialization routines automatically included by $XPSLINK as overlay segments.

Before including this statement in your link-control data set, see "Define an Overlay Area" on page IS-63.

INCLUDE Statement

The INCLUDE statement identifies each supervisor object module to be included in your generated supervisor.

An explanation of each supervisor object module and the criteria for including a module in your supervisor follows. The modules are shown in the same order as they appear in work sheet 3.

Supervisor Support

These supervisor object modules ***must*** reside in partition 1.

EDXSYS EDXSYS is a required module and must be included in every supervisor. EDXSYS contains the following five tables:

- Reserved storage locations
- Device vector table
- Communications vector table
- Task supervisor work area
- Emulator command table.

ASMOBJ ASMOBJ is a required module and must be included in every supervisor. This module will contain the object code resulting from assembling the $EDXDEF data set which contains the system definition statements. You have the option of changing the name of this module or use it as it exists.

EDXSVCX EDXSVCX is a required module and must be included in every supervisor. It performs task requests.

Select Your Required Support

Step 3 - Selecting Your Software Support *(continued)*

$DBUGNUC	$DBUGNUC must be included if you wish to debug programs using the $DEBUG utility.
EDXALU	EDXALU is a required module and must be included in every supervisor.
EDXSTART	EDXSTART is a required module and must be included in every supervisor. EDXSTART includes the system initialization task. It is attached at IPL and initializes the supervisor by calling in EDXINIT which processes the initialization routines. EDXSTART is used also for default exception processing for program, machine, and soft exception checks.
SWAITM	SWAITM is an optional module and is required if you use the wait on multiple events option. If you include SWAITM, be sure to code the MECBLIST operand on the SYSTEM statement.

Timer Support

These supervisor object modules ***must*** be located in in partition 1.

EDXTIMER	EDXTIMER must be included if your Series/1 has a 4955 processor and you defined the #7840 timer feature with the TIMER definition statement. If you include EDXTIMER, $XPSLINK automatically includes the initialization module TIMRINIT.
EDXTIMR2	EDXTIMR2 must be included if your Series/1 is a 4952, 4954, or 4956 processor with native timer support. If you include EDXTIMR2, $XPSLINK automatically includes the initialization module CLOKINIT.

EXIO Control Support

This supervisor object module ***must*** be located in in partition 1.

IOSEXIO	IOSEXIO is optional and is required only if you intend to attach and control OEM devices or to support standard devices in a nonstandard manner. The device is defined with the EXIODEV definition statement. If you include IOSEXIO, $XPSLINK automatically includes the initialization module EXIOINIT.
EXIOTRC	EXIOTRC is optional and is required only if you want to trace I/O operations and interrupts for an EXIO device with the $TRACEIO utility. In addition, EXIOTRC enables you to record conditions codes, completion codes, ISB and DCB information, and status words for a device interrupt from a start-cycle-steal-status (SCSS) operation. If you do not include EXIOTRC and include IOSEXIO, $XPSLINK automatically includes NOEXIOTR.

Step 3 - Selecting Your Software Support *(continued)*

Error Logging Support

These supervisor object modules ***must*** be located in partition 1.

SYSLOG	SYSLOG is used by the $LOG utility to log errors to a disk or diskette data set. $XPSLINK automatically includes SYSLOG.
CIRCBUFF	CIRCBUFF is required if the storage program check/machine check log is to be kept. CIRCBUFF implements the optional software trace table. This table contains program check, soft exception, and machine check trace data. It also offers information on the types of errors that have been occurring. (Refer to the *Problem Determination Guide* for more information.)

Optional Function Support

These supervisor object modules ***must*** be located in in partition 1.

RLOADER	RLOADER is required if you intend to load programs from secondary storage (disk or diskette) with either the $L operator command or the LOAD instruction. If you include RLOADER, $XPSLINK automatically includes LOADINIT. If you do not include RLOADER, you must link edit your application programs with the supervisor for them to execute (see $PROG1 under "System Initialization Support" on page IS-60 for additional information).
STORMGR	STORMGR is required to gain access to unmapped storage from an application program or to use the $MEMDSK utility to set up a memory disk volume. The $MEMDSK utility allows you to use part or all of unmapped storage as if it were a single-volume disk. See the *Operator Commands and Utilities Reference* for information on the $MEMDSK utility. If you include STORMGR, $XPSLINK automatically includes STORINIT. If you do not include STORMGR, $XPSLINK automatically includes NOSTGMGR. STORMGR contains support for the Event Driven Language instructions GETSTG, FREESTG, and SWAP, and the STORBLK statement. STORBLK creates an unmapped storage control block, and GETSTG, FREESTG, and SWAP enable application programs to use unmapped areas in the system.
IAMQCB	IAMQCB is required if the Indexed Access Method Version 2 (5719-AM4) is installed or will be installed in the future.
PWRAM80	PWRAM80 is an optional module that can be included for the 4980 display station. PWRAM80 automatically loads the control store, image store, and microcode necessary for 4980 display station operation. If you do not include PWRAM80 and you turn off a 4980 display station, you must IPL your system to make the 4980 operational again.

Step 3 - Selecting Your Software Support *(continued)*

Host Communication Facility (HCF)

This supervisor object module ***must*** be located in partition 1.

TPCOM	TPCOM is required if you defined host communication support (communication to a S/370) with the HOSTCOMM definition statement. If you include TPCOM, $XPSLINK automatically includes the initialization module TPINIT.

Translation Table Support

These supervisor object modules ***must*** be located in partition 1.

TRASCII	TRASCII is required if feature numbers #1310, #2095/2096, and #7850 are used to connect ACCA or TTY I/O devices to your Series/1. **Note:** You must include this module or TREBASC as well as both IOSTERM and IOSACCA for the Remote Support Link. Refer to *Problem Determination Guide* for more information about the Remote Support Link.
TREBASC	TREBASC is required if feature numbers #1610 and #2091/2092 are used to connect ACCA or TTY I/O devices to your Series/1. **Note:** You must include this or TRASCII as well as both IOSTERM and IOSACCA for the Remote Support Link. Refer to *Problem Determination Guide* for more information about the Remote Support Link.
TREBCD or **TRCRSP**	Include either of these modules if you defined a 2741 terminal (DEVICE=2741) to the supervisor. A 2741 terminal can use either TREBCD or TRCRSP. Include TREBCD for processor-to-processor support (DEVICE=PROC).

Disk(ette) and Tape Support

These supervisor object modules can be placed in ANY partition. However, any modules that you include ***must*** remain as a group.

DISKIO	DISKIO must be included if you defined a disk or diskette device using the DISK definition statement. If you include DISKIO, DISKINIT and DSKINIT2 are included automatically.
D49624	D49624 is optional and is required only if you defined a 4962 disk and/or 4964 diskette with the DISK definition statement.
D4963A	D4963A is optional and is required only if you defined a 4963, 4967, DDSK-30, or DDSK-60 disk with the DISK definition statement.
D4966A	D4966A is optional and is required only if you defined a 4965 and/or 4966 diskette unit with the DISK definition statement.

D1024	D1024 must be included if you intend to use double density diskettes at 1024 bytes per sector on a 4965 or 4966 diskette unit. Once included, a copy of $IO1024 is loaded automatically for each disk task defined for a 4965 and/or 4966 diskette unit. $IO1024 is loaded into the highest available partition the first time 1024 bytes per sector support is required for a particular device. $IO1024 remains in storage until that diskette unit is varied offline or is varied online again with a non-1024-bytes-per-sector diskette.
D4969A	D4969A is required if you defined one or more 4968 or 4969 tape units with the TAPE definition statement. If you include D4969A, $EDXLINK automatically includes the initialization module TAPEINIT.

Terminal Support

These supervisor object modules can be placed in any partition. However, any modules that you include ***must*** remain as a group.

EDXTIO	EDXTIO must be included if you defined terminals with the TERMINAL definition statement. Terminals include the 4973, 4974, 4975, 4975-01A 4978, 4979, 4980, 5219, 5224, 5225, GPIB, Series/1-Series/1, Virtual terminals, ACCA, TTY, 2741, and 4013. If you include EDXTIO, $XPSLINK automatically includes the initialization modules TERMINIT and EDXTERMQ. If you do not include EDXTIO, $XPSLINK automatically includes NOTIO.
MINMSG	MINMSG must be included if you wish to have only that part of the messages that are resident in storage appear. This means that only the message text is issued; parameters are not issued. If system commands and/or utilities are used, the messages will not contain the parameters and may not be meaningful. For both systems with secondary storage (disk or diskette) and without secondary storage, MINMSG is the only module required for storage-resident messages support. If you include MINMSG, do not include FULLMSG.
FULLMSG	FULLMSG must be included if you wish to have full message data set support. This means that messages (supervisor, initialization, and utility messages) are shown as full messages (message number and text rather than just message number). FULLMSG should be included if system commands and/or utilities are to be used on your system. If you include FULLMSG, do not include MINMSG.
IOS3101	IOS3101 is required if you are using a 3101 in block mode (DEVICE=ACCA,MODE=3101B). If you include this module, you must include IOSACCA as well.
IOS4974	IOS4974 is optional and is required only if you defined a 4973, 4974, 4975 (except 4975-01A), 5219, 5224 and/or 5225 printer using the TERMINAL definition statement.

Step 3 - Selecting Your Software Support *(continued)*

IOS4975A	IOS4975A is optional. Include it only if you defined a 4975-01A ASCII printer using the TERMINAL definition statement. If you include this module, $XPSLINK automatically includes IOSTERM and IOSACCA.
IOS4979	IOS4979 is optional and is required only if you defined a 4978, 4979, and/or 4980 display station using the TERMINAL definition statement. If you included the OPTION NOVERLAY statement, you must include the initialization modules for the 4978, 4979, and 4980 display stations. For the 4978 and 4979, include INIT4978; for the 4980, include INIT4980.
IOSACCA	IOSACCA is required if you defined an ACCA-type terminal with the TERMINAL definition statement. For terminals defined as ACCA, the appropriate translation tables must be included (see Translation Tables Support). **Note:** You must include this module for the Remote Support Link. Refer to *Problem Determination Guide* for more information about the Remote Support Link.
ACCATRC	ACCATRC is optional and is only required if you want to trace I/O operations and interrupts on an ACCA line with the $TRACEIO utility. In addition, ACCATRC enables you to record condition codes, completion codes, ISB and DCB information, and status words for a device interrupt from a start cycle steal status (SCSS) operation. If you do not include ACCATRC but you do include IOSACCA, $XPSLINK automatically includes NOACCATR.
IOSTERM	IOSTERM is required if you defined any of the following devices with the TERMINAL definition statement: • ACCA-type terminal • 3101 in character mode • Tektronics 40xx. **Note:** You must include this module for the Remote Support Link. Refer to *Problem Determination Guide* for more information about the Remote Support Link.
IOSTTY	IOSTTY is optional and provides support for ASR 33/35/3101/3101C TTY devices. Include it only if you defined DEVICE=TTY with the TERMINAL definition statement. If you include this module, $XPSLINK automatically includes IOSTERM.
IOS2741	IOS2741 is optional and provides support for 2741 terminals. Include it only if you defined DEVICE=2741 with the TERMINAL definition statement. If you include this module, $XPSLINK automatically includes IOSTERM.

IOS4013	IOS4013 is optional and provides support for the Tektronix 4000 Series of display terminals. Include it only if you defined DEVICE=4013 with the TERMINAL definition statement. If you include IOS4013, $XPSLINK automatically includes IOSTERM and the initialization module INIT4013.
IOSGPIB	IOSGPIB is optional and provides support for the General Purpose Interface Bus. Include it only if you defined DEVICE=GPIB with the TERMINAL definition statement.
IOSS1S1	IOSS1S1 is optional and provides support for the Series/1-to-Series/1 Attachment. Include it only if you defined DEVICE=S1S1 with the TERMINAL definition statement. If you include IOSS1S1, $XPSLINK automatically includes the initialization module S1S1INIT.
IOSVIRT	IOSVIRT is optional and provides support for virtual terminal communications. Include it only if you defined DEVICE=VIRT with the TERMINAL definition statement. IOSVIRT must be included if you intend to use the Remote Management Utility with the PASSTHRU function.
IOSPOOL	IOSPOOL is optional and needs to be included only if printer spooling is to be used.
EBFLCVT	EBFLCVT is required for applications that use EBCDIC/floating-point conversion and the GETEDIT, PUTEDIT, FORMAT, FPCONV, CONVTD, or CONVTB instructions. EBFLCVT includes the routines that convert EBCDIC values to floating-point values and floating-point values to EBCDIC values. If you include this module, include EDXTIO.

Floating Point Support

This supervisor object module can be placed in any partition.

EDXFLOAT	EDXFLOAT is required for applications using floating-point data manipulation instructions in FORTRAN, Pascal, PL/I or EDL. EDXFLOAT processes the Event Driven Language floating-point arithmetic instructions. These instructions require the floating-point hardware feature. If you do not include EDXFLOAT, $XPSLINK automatically includes NOFLOAT. NOFLOAT causes an exception condition if a program attempts to execute a floating-point instruction.

Queue I/O Support

This module can be placed in any partition.

QUEUEIO	QUEUEIO is required for queueing operations which are applications that use the FIRSTQ, NEXTQ, LASTQ, and DEFINEQ statements.

Step 3 - Selecting Your Software Support *(continued)*

Binary Synchronous Communications Support

These supervisor object modules can be placed in any partition.

BSCAM	BSCAM is required if you defined a binary synchronous communication line using the BSCLINE definition statement. If you include BSCAM, $XPSLINK automatically includes the initialization module BSCINIT.
BSCX21	BSCX21 is required to support the X.21 switched network support with binary synchronous communication support. To specify retries and delay time, you must also include timer support in your supervisor.

Sensor I/O Support

These supervisor object modules can be placed in any partition. However, if they are left in partition 1, it improves performance.

SBCOM	SBCOM is required to support sensor I/O devices (AI, AO, DI, DO, or PI) defined with the SENSORIO definition statement. If you include SBCOM, $XPSLINK automatically includes IOLOADER and the initialization module SBIOINIT.
SBAI	SBAI is optional and is required if you defined analog input support with the SENSORIO definition statement.
SBAO	SBAO is optional and is required if you defined analog output support with the SENSORIO definition statement.
SBDIDO	SBDIDO is optional and is required if you defined digital input/output support with the SENSORIO definition statement.
SBPI	SBPI is optional and is required if you defined process interrupt support with the SENSORIO definition statement.

System Initialization Support

These supervisor object modules ***must*** be located in partition 1.

$PROG1 $PROG1 is required if you intend to link edit a single application program with the supervisor to form a single load module. By doing this, the application program will always be resident in storage and will be started automatically after the system and user-written initialization is complete. Using $PROG1 can be useful if your system does not have disk or diskette devices from which to load programs. Refer to the *Customization Guide* for information on writing a program using $PROG1.

To remove the execution of a disk(ette)-resident program feature from your supervisor, do not include RLOADER. If the system-resident disk or diskette volume (normally EDX002) exists on the system, delete the transient loader ($LOADER) from it.

IO1024	IO1024 is optional and is required only if you intend to IPL your system from a 1024-bytes/sector diskette. It must be included before EDXINIT.

System Support - Initialization

These modules ***must*** be located in partition 1.

EDXINIT	EDXINIT is a required module and must be included in every supervisor. It must appear in this location. EDXINIT is the system initialization routine and executes all the initialization modules. **Note:** If you intend to add your own operator command, user attention list, or device support, the INCLUDE statement and module name must be placed before EDXINIT. For information on adding your own operator command, see *Customization Guide.*
RW4963ID	RW4963ID is required to initialize a 4963 disk with fixed-head support and is defined with the DISK definition statement. Include this module only if you specify OPTION NOVERLAY.
INITADAP	INITADAP is required to initialize the Printer Attachment - 5200 Series (feature #5640) and the Multidrop Work Station Attachment (feature #1250) defined with the ADAPTER statement. Include this module only if you specify OPTION NOVERLAY.
INITMFA	INITMFA is required to initialize a Multifunction Attachment Feature #1310 defined with the ADAPTER definition statement. Include this module only if you specify OPTION NOVERLAY.
INIT4978	INIT4978 is required to initialize the 4978 and 4979 display terminals defined with the TERMINAL definition statement. Include this module only if you specify OPTION NOVERLAY.
INIT4980	INIT4980 is required to initialize the 4980 display station defined with the TERMINAL definition statement. Include this module only if you specify OPTION NOVERLAY.

Step 3 - Selecting Your Software Support *(continued)*

User-Written Initialization Modules

You may want to write your own device-initialization routines. If you do, you can include them at this location or preceding the EDXINIT module. However, if you include them preceding the EDXINIT module, the initialization routines will remain resident in your supervisor. For information on writing your own initialization routines, see the *Customization Guide*.

In addition, you have the option of defining your own device-initialization routines as overlay segments. An overlay segment consists of an OVERLAY statement and the INCLUDE statements associated with it. Each segment is ended by the next OVERLAY statement or PART statement. Generally, each initialization routine is a separate overlay segment (one OVERLAY statement and one INCLUDE statement). However, you can define multiple routines in one overlay segment if you wish. See the $EDXLINK utility in the *Operator Commands and Utilities Reference* for an explanation of the OVERLAY statement.

If you do not want to define your device-initialization routines as overlay segments, specify only the INCLUDE statements for each initialization routine.

Object Modules to be Located Outside of Partition 1

If you are going to generate a multipartition supervisor, you must locate object modules outside of partition 1. The PART statement preceding a group of modules specifies the partition where the modules are to be located. Each group of modules to be located outside of partition 1 must be moved in the $LNKCNTL data set to this location.

LINK Statement

The LINK statement is a required link-control statement used to define the name of the data set where the generated supervisor is to be stored. The name of the data set is $EDXNUCT in the $LNKCNTL data set. As shown in work sheet 3, the name of the supervisor is part of the $XPSLINK LINK statement and MUST be specified. The generated supervisor will be stored on EDX002. The REPLACE option causes the linkage editor to replace any program on the same volume with the same name as the one specified in the LINK statement. The END option ends the linkage editor operation upon completion of LINK.

The $EDXNUCT data set is allocated automatically by $XPSLINK. You may wish to change this name to $EDXNUCx (x = any alphameric character) to save different supervisor versions in individual data sets. However, if you do change the data set name, be sure the supervisor name starts with the seven characters $EDXNUC and that the name is different from the name of the current supervisor.

Define an Overlay Area

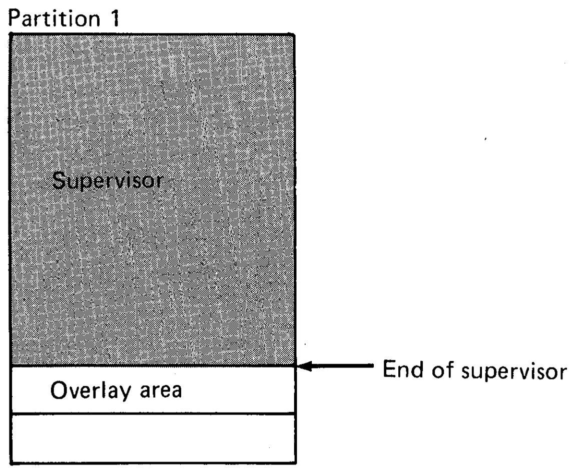
An overlay area is an area in storage that is used to execute overlay segments. There are two ways to create an overlay area in your operating system:

1. You can allow the supervisor to create the overlay area for you. The supervisor creates the overlay area directly after the supervisor area in partition 1. This is the default.

   Example: The supervisor program creates the overlay area. In the link-control data set, the OPTION NOVERLAY and OVLAREA statements must be commented out with an asterisk (*).

```
*OPTION NOVERLAY        *25*  NO OVERLAY STRUCTURE
*-----------------------------------------------------------------
*SUPERVISOR SUPPORT       - MUST BE FIRST AND IN PARTITION 1
*-----------------------------------------------------------------
 PART1
 VOLUME XS5002                DEFAULT VOLUME FOR INCLUDE MODULES
*OVLAREA OVLSTART OVLEND *23* USER DEFINED OVERLAY AREA
 INCLUDE EDXSYS          *1*  SYSTEM TABLES AND WORK AREAS
     .
     .
*******************************************************************
* SYSTEM SUPPORT -- INITIALIZATION
*******************************************************************
 INCLUDE EDXINIT         *24* SUPERVISOR INITIALIZATION
*
*******************************************************************
```

   The overlay area created by the supervisor appears in storage as follows:



Step 3 - Selecting Your Software Support *(continued)*

2. You can define the overlay area by including the OVLAREA statement in the link-control data set.

   The OVLAREA statement defines the overlay area within the storage in partition 1 occupied by the supervisor. OVLSTART indicates the starting address of the overlay area. OVLSTART is a pointer to the beginning of the SEGINIT module within the supervisor which is equated to DISKBUFR+12 bytes. OVLEND indicates the end of the overlay area. OVLEND is a pointer to the end of the DISKINIT module within the supervisor.

   However, you can specify any starting and ending address for the overlay area, but the size of the overlay area must be larger than the largest overlay segment included. If the overlay area is not large enough, $XPSLINK issues the following message:

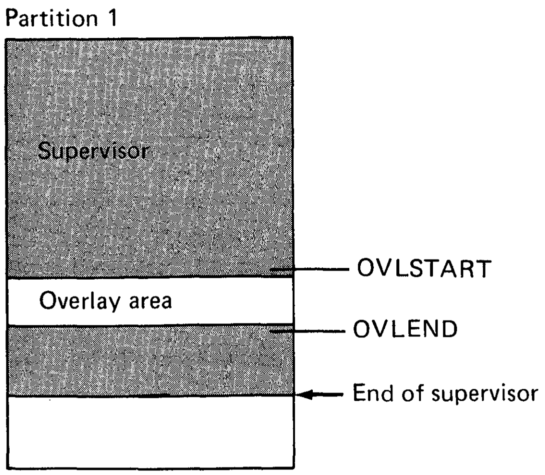
```
OVERLAY AREA IS TOO SMALL, SIZE REQUIRED IS xxxx HEX
```

   where xxxx indicates the required size of the overlay area in hexadecimal. Respecify the starting and ending points of the overlay area and link edit the definition statements with the link-control data set.

   Example: Define the overlay area to be within the storage in partition 1 occupied by the supervisor. In this case, remove the asterisk (*) from the OVLAREA statement and leave the asterisk (*) on the OPTION NOVERLAY statement.

```
*OPTION NOVERLAY       *25*  NO OVERLAY STRUCTURE
*-----------------------------------------------------------------
*SUPERVISOR SUPPORT      - MUST BE FIRST AND IN PARTITION 1
*-----------------------------------------------------------------
 PART1
 VOLUME XS5002               DEFAULT VOLUME FOR INCLUDE MODULES
 OVLAREA OVLSTART OVLEND *23* USER DEFINED OVERLAY AREA
 INCLUDE EDXSYS          *1*  SYSTEM TABLES AND WORK AREAS
     .
     .
*******************************************************************
* SYSTEM SUPPORT -- INITIALIZATION
*******************************************************************
 INCLUDE EDXINIT         *24* SUPERVISOR INITIALIZATION
*
*******************************************************************
```

The overlay area appears in storage as follows:



In both cases, an area of storage in partition 1 is defined as an overlay area. The overlay area is large enough to contain the largest initialization routine.

Including Initialization Routines in Your Supervisor

Initialization routines are supervisor object modules that prepare I/O devices for operation. The linkage editor, $XPSLINK, automatically includes the initialization routines required for your system configuration in your supervisor. However, by excluding or including certain statements in the link-control data set, initialization routines are treated either as overlay segments or as resident programs.

Initialization Routines as Overlay Segments

To include initialization routines as overlay segments, you do not need to include any statements in the link-control data set. The linkage editor builds the supervisor to default to an overlay structure. As a result, $XPSLINK includes the required initialization routines as overlay segments. This means that each initialization routine is executed one at a time in the storage defined as the overlay area.

Initialization Routines as Resident Programs

To include initialization routines as resident programs, you must include the OPTION NOVERLAY statement. The linkage editor builds the supervisor without an overlay structure and automatically includes the initialization routines required for your system configuration as resident programs. In this case, the required initialization routines are loaded simultaneously in supervisor storage at IPL time. As a result, the initialization routines increase the amount of storage that the supervisor requires in partition 1.

Step 3 - Selecting Your Software Support *(continued)*

If you include the OPTION NOVERLAY statement when tailoring a system for use on a 4952, 4954, 4955, or 4956 processor, the following five initialization routines are not included automatically by the linkage editor:

- RW4963ID
- INITMFA
- INITADAP
- INIT4978
- INIT4980.

You must include each of the above modules specifically if you require them as part of the supervisor.

The RW4963ID, INITMFA, INITADAP, INIT4978, and INIT4980 modules do not appear in the link-control data set, but do appear in work sheet 3. You must add them to the link-control data set directly following the EDXINIT module.

Example: Include the OPTION NOVERLAY statement and the initialization modules that are not included automatically by the linkage editor (OVERLAY is the default).

```
 OPTION NOVERLAY       *25* NO OVERLAY STRUCTURE
*-----------------------------------------------------------------
*SUPERVISOR SUPPORT       - MUST BE FIRST AND IN PARTITION 1
*-----------------------------------------------------------------
 PART1
 VOLUME XS5002              DEFAULT VOLUME FOR INCLUDE MODULES
*OVLAREA OVLSTART OVLEND *23* USER DEFINED OVERLAY AREA
 INCLUDE EDXSYS          *1* SYSTEM TABLES AND WORK AREAS
    .
    .
*******************************************************************
* SYSTEM SUPPORT -- INITIALIZATION
*******************************************************************
 INCLUDE EDXINIT         *24* SUPERVISOR INITIALIZATION
 INCLUDE RW4963ID        *3*  4963 FIXED HEAD REFRESH SUPPORT
 INCLUDE INITMFA         *3*  MFA INITIALIZATION
 INCLUDE INITADAP        *3*  ALPA & SMIO INITIALIZATION
 INCLUDE INIT4978        *3*  4978 DISPLAY INITIALIZATION
 INCLUDE INIT4980        *3*  4980 DISPLAY INITIALIZATION
*
*******************************************************************
```

Initialization Modules

The following is a list of the initialization modules that are included in your supervisor, if required, by the linkage editor, along with an explanation of each. This list is provided for information only as the initialization module names may appear in the link map when you generate your system.

Module	Description
BSCINIT	BSCINIT is required to initialize a binary synchronous line defined with the BSCLINE definition statement.
CLOKINIT	CLOKINIT is required to initialize the native timer on the 4952, 4954, or 4956 processors.
DSKINIT2	DSKINIT2 is required to initialize fixed-head disk device(s) defined with the DISK definition statement, set attention for 1024-bytes-per-sector diskettes, and issue the IPL volume message.
EXIOINIT	EXIOINIT is required to initialize I/O devices defined with the EXIODEV definition statement.
INITADAP	INITADAP is required to initialize the Printer Attachment Feature #5640 and the Multidrop Work Station Attachment Feature #1250 defined with the ADAPTER definition statement.
INITMFA	INITMFA is required to initialize the Multifunction Attachment Feature #1310 (MFA) defined with the ADAPTER definition statement.
INIT4013	INIT4013 is required to initialize a Tektronics 4013 Graphics Terminal defined with the TERMINAL statement.
INIT4978	INIT4978 is required to initialize 4978 and 4979 display terminals defined with the TERMINAL definition statement.
INIT4980	INIT4980 is required to initialize 4980 display terminals defined with the TERMINAL definition statement.
LOADINIT	LOADINIT is required if programs are to be loaded from disk or diskette.
RW4963ID	RW4963ID is required to initialize a 4963 disk with fixed-head support and is defined with the DISK statement. You must include DSKINIT2 in the same overlay area.
SBIOINIT	SBIOINIT is required to initialize sensor-based I/O devices defined with the SENSORIO definition statement.

Step 3 - Selecting Your Software Support *(continued)*

STORINIT	STORINIT is required to initialize the unmapped storage tables in your system if you have defined unmapped storage.
S1S1INIT	S1S1INIT is required to initialize the Series/1-to-Series/1 attachment defined with the TERMINAL definition statement. IOSS1S1 must be included (see Terminal Support).
TAPEINIT	TAPEINIT is required to initialize a 4968 or 4969 tape unit defined with the TAPE definition statement.
TERMINIT	TERMINIT is required to initialize any terminals defined with the TERMINAL definition statement.
TIMRINIT	TIMRINIT is required to initialize the #7840 timer on the 4955 processor.
TPINIT	TPINIT is required to initialize the Host Communications Facility defined with the HOSTCOMM definition statement.

Step 4 - Defining Supervisor Structure

In "Step 1 - Estimating Total Supervisor Size" on page IS-42, you estimated the total size of your supervisor. If your supervisor size is 64K or greater or you want to free up storage in partition 1 in order to have space available for additional I/O devices or program execution, you need to reduce the size of the supervisor in partition 1.

Reducing the Size of Your Supervisor

There are three ways to reduce the size of your supervisor:

1. The first way is to minimize the amount of storage in partition 1 used by the initialization routines by treating them as overlay segments. An initialization routine is an EDX program used to set up the necessary internal controls to make a device active. An overlay segment is a portion of a program that is read from disk, executed, and released.

   To do this, an area of storage within partition 1 is defined as an overlay area and is large enough to contain the largest initialization routine. At IPL time, each initialization routine is brought into storage one at a time in the same storage area thereby reducing the amount of storage required.

2. The second way is to break your supervisor into several parts. Your supervisor is a program that is made up of many smaller programs called object modules. Some of these modules can be moved to partitions other than partition 1. If you move part of the supervisor to another partition, you reduce the size of the supervisor remaining in partition 1. For example, if your supervisor is approximately 48K bytes in size and you move several parts

that equal 8K bytes in size, the size of the supervisor remaining in partition 1 is 40K (48K - 8K = 40K).

By locating the supervisor in more than one partition, you need less storage for it in partition 1. This leaves more storage available in partition 1 for support of additional I/O devices or to execute application programs within partition 1. However, by placing parts of the supervisor in other partitions, the amount of storage within those partitions available for program execution is reduced.

3. The third way is a combination of (1) and (2) above; that is, treat initialization routines as overlay segments and spread the supervisor across more than one partition.

Any storage within each partition not containing part of the supervisor is available for execution of application programs, program products, and utility programs. Depending upon your processor type, your supervisor can support processor storage sizes from 64K bytes (1K = 1024 bytes) to 1024K bytes with unmapped storage support.

In the SYSTEM definition statement, you define the partition structure and the amount of storage assigned to each partition through the PART= operand. You also define the number of programs that can execute concurrently in each partition through the MAXPROG= operand. In order to define these characteristics, you need to make a couple of decisions:

1. What portion of each partition will be used by the supervisor.

2. How much storage is left within each partition for the execution of programs.

To help you make these decisions, use work sheet 4 in Appendix E. Work sheet 4 is made up of two parts: A and B.

Part A - Estimate Storage Needed by Supervisor Object Modules

Part A lists the approximate sizes (in bytes) of the supervisor object modules. Under 'PARTITION ONE' is a list of modules that ***must*** reside in partition one. Under 'PARTITION ANY' are the names of modules that can be placed in partitions 2 through 8 (depending upon your processor). However, some of these modules ***must*** be grouped together. These subgroupings are shown on the work sheet.

By determining which modules you want to place in partitions 2 through 8, you can estimate how much storage these modules will occupy in a partition. The storage remaining in each partition is available for other programs. In addition, by totaling the number of bytes of these modules, you can estimate the reduction in size of that part of the supervisor remaining in partition 1.

Step 4 - Defining Supervisor Structure *(continued)*

Part B - Estimate Storage Needed by Other Programs

Part B lists the amount of storage required by the supervisor to support program products, application programs, and utility programs. These programs, although not part of your supervisor, use storage within the partition where they are loaded. To determine how many programs you can execute concurrently within a partition, you need to know how much storage is required by each program. An estimate of more than 64K or 65536 bytes for a specific program indicates that it will not be able to execute within one specific partition.

The approximate size (in bytes) of each system utility program is shown. You may or may not use all of the utility programs. However, you should make sure that your partitions are large enough to hold the largest utility program you plan to use along with other programs.

After you determine which object modules you are moving out of partition 1, the amount of storage left in each partition for program execution, and the maximum number of programs you can execute within each partition, you may need to redefine the following:

Work Sheet 2	The PARTS= and MAXPROG= operands of the SYSTEM definition statement.
Work Sheet 3	The PART link-control statement indicating which modules are being moved from partition 1 and the partition where they will be located.

Work Sheets for the Sample System

Following are samples of work sheets 2 and 3 that reflect the hardware and software features of the sample system defined in the beginning of the chapter.

"Work Sheet 2 for Sample System" on page IS-71 shows the definition statements defining the devices and "Work Sheet 3 for Sample System" on page IS-73 shows the software support selected.

IBM Series/1 Event Driven Executive

Work Sheet 2—Installation and System Generation

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Storage definition				
blank	SYSTEM	MAXPROG=	(1 , 10 , 10 , 5 , 5 , ____ , ____ , ____),	Optional-defaults to 10
		PARTS=	(0 , 24 , 16 , 20 , 32 , ____ , ____ , ____),	Optional-defaults to 32K
		DATEFMT=	____ ,	Optional-defaults to MMDDYY
		IABUF=	____ ,	Optional-defaults to 20
		XPSSTK=	____ ,	Optional-defaults to 20
		COMMON=	(0 , 5 , 0 , 2 , 3 , ____ , ____ , ____),	Optional-defaults to EDXSYS
		INITPRT=	____ ,	Optional-defaults to 1
		INITMOD=	(____ , ____ , ____ , ____)	Optional
		MECBLST=	____	Optional
Disk(ette) definition				
blank	DISK	DEVICE=4962,-1F		Required
		ADDRESS=	03 ,	Required
		VOLNAME=	(____ , ____ , ____ , ____),	Optional
		TASK=	YES ,	Optional-defaults to NO
		END=	____	Optional-defaults to NO
blank	DISK	DEVICE= 4964,		Required
		ADDRESS=	02 ,	Required
		TASK=	YES ,	Optional-defaults to NO
		END=	YES	Optional-defaults to NO
4973/4974 Printers				
~~label~~ $SYSPRTR	TERMINAL	DEVICE=	4973	Required (specify 4973 or 4974)
		ADDRESS	21 ,	Required
		PAGSIZE=	____ ,	Optional
		LINSIZE=	____ ,	Optional
		TOPM=	____ ,	Optional
		BOTM=	____ ,	Optional
		LEFTM=	____ ,	Optional
		RIGHTM=	____ ,	Optional
		OVFLINE=	____ ,	Optional-defaults to NO
		SPOOL=	____ ,	Optional-defaults to NO
		END=	____	Optional-defaults to NO (Specify YES if last TERMINAL statement)

4978/4979 Terminals

~~label~~ $SYSLOG	TERMINAL	DEVICE=	4979,	Required (specify 4978 or 4979)
		ADDRESS=	04,	Required
		PAGSIZE=	______,	Optional
		LINSIZE=	______,	Optional
		TOPM=	______,	Optional
		BOTM=	______,	Optional
		NHIST=	______,	Optional
		LEFTM=	______,	Optional
		RIGHTM=	______,	Optional
		OVFLINE=	______,	Optional-defaults to NO
		HDCOPY=	$SYSPRTR,	Optional
		ATTN=	______,	Optional-defaults to YES
		PF1=	______,	Optional-(2 digit hex char)
		SCREEN=	______,	Optional-(YES/NO/STATIC)
		PART=	______,	Optional-defaults to partition 1
		SPOOL=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO (Specify YES if last TERMINAL statement)

2ND 4979

```
$SYSLOGA TERMINAL  DEVICE=4979,
                   ADDRESS=24,
                   HDCOPY=$SYSPRTR,
                   END=YES
```

System common

$SYSCOM	CSECT		
	~~QCB~~	NONE DEFINED	
	~~QCB~~		
	~~ECB~~		
	~~ECB~~		
	ENTRY	$EDXPTCH	
$EDXPTCH	DATA	128F'0'	System Patch Area

Work Sheets for the Sample System *(continued)*

Work Sheet 3 for Sample System

IBM Series/1 Event Driven Executive

Work Sheet 3—Installation and System Generation

To include a supervisor object module in your operating system, leave Column One blank. To exclude a module, place an asterisk (*) in Column One.

Column one	Supervisor object modules	Notes	Purpose of module
*	OPTION NOVERLAY	*25*	No overlay structure
Supervisor support—must be first and in partition 1			
	PART 1		
	VOLUME SX4002		Default volume for include modules
*	OVLAREA OVLSTART OVLEND	*23*	User defined overlay area
	INCLUDE EDXSYS	*1*	System tables and work area
	INCLUDE ASMOBJ,EDX002	*1*	Output from user system generation
	INCLUDE EDXSVCX	*1*	Task supervisor
*	INCLUDE $DEBUGNUC	*2*	Resident $DEBUG support
	INCLUDE EDXALU	*1*	EDL instruction emulator
	INCLUDE EDXSTART	*1*	Initialization & error handler
*	INCLUDE SWAITM	*3*	Wait on multiple events
Timer support—must be in partition 1			
*	INCLUDE EDXTIMER	*12*	4955 timer (7840) support
*	INCLUDE EDXTIMR2	*12*	4952/4954/4956 timer support
EXIO support—must be in partition 1			
*	INCLUDE IOSEXIO	*3*	EXIO device control support
*	INCLUDE EXIOTRC	*3*	EXIO trace option
Error logging—must be in partition 1			
*	INCLUDE SYSLOG	*3*	I/O error logging
*	INCLUDE CIRCBUFF	*16*	Program/machine check logging
Optional function support—must be in partition 1			
*	INCLUDE RLOADER	*17*	Relocating program loader
*	INCLUDE STORMGR	*3*	Unmapped storage manager support
*	INCLUDE IAMQCB	*20*	IAMQCB needed for IAM support
*	INCLUDE PWRAM80	*26*	4980 Power on RAM
Host communications support—must be in partition 1			
*	INCLUDE TPCOM	*14*	Host communication support
Translation tables—must be in partition 1			
*	INCLUDE TRASCII	*10*	1310,2095/2096,7850 ACCA/TTY translation
*	INCLUDE TREBASC	*10*	1610,2091/2092 ACCA translation
*	INCLUDE TREBCD	*11*	2741,PROC EBDC translation
*	INCLUDE TRCRSP	*11*	2741 correspondence translation
Disk(ette) and tape support—must be grouped together in any partition			
	PART 1		
	INCLUDE DISKIO	*3*	Basic disk(ette) support
	INCLUDE D49624	*3*	4962/4964 disk(ette) support
*	INCLUDE D4963A	*3*	4963/4967/DDSK-30 subsystem support
*	INCLUDE D4966A	*3*	4965/4966 disk(ette) support
*	INCLUDE D1024	*3,21*	1024 bytes/sector IO support
*	INCLUDE D4969A	*3*	Basic tape support
Terminal—must be grouped together in any partition			
	PART ~~1~~ 8		
	INCLUDE EDXTIO	*4*	Basic terminal support
*	INCLUDE MINMSG	*5*	Message ID only support
	INCLUDE FULLMSG	*5*	Message data set support
*	INCLUDE IOS3101	*7*	ACCA 3101B support
	INCLUDE IOS4979	*3*	4978/4979/4980 display support

Work Sheets for the Sample System *(continued)*

IBM Series/1 Event Driven Executive

Work Sheet 3—Installation and System Generation (cont.)

Column one	Supervisor object modules	Notes	Purpose of module
	INCLUDE IOS4974	*3*	4973/4974/4975/5219/5224/5225 printer support
*	INCLUDE IOSACCA	*4*	ACCA device handler
*	INCLUDE ACCATRC	*3*	ACCA trace option
*	INCLUDE IOSTERM	*A,6*	ACCA/TTY/2741/4013 support
*	INCLUDE IOSTTY	*3*	ASR 33/35/3101/3101C TTY support
*	INCLUDE IOS2741	*3*	2741 terminal support
*	INCLUDE IOS4013	*3*	Digital I/O terminal support
*	INCLUDE IOSGPIB	*3*	GPIB support
*	INCLUDE IOSS1S1	*3*	Series/1 - Series/1 support
*	INCLUDE IOSVIRT	*3,9*	Virtual terminal support
*	INCLUDE IOSPOOL	*3*	Spooling support
*	INCLUDE EBFLCVT	*18*	EBCDIC/floating point conversion
Floating point—may be in any partition			
*	PART 1		
*	INCLUDE EDXFLOAT	*3*	Floating point arithmetic
Queue I/O—may be in any partition			
*	PART 1		
*	INCLUDE QUEUEIO	*19*	Queue processing support
Bisync communications—may be in any partition			
*	PART 1		
*	INCLUDE BSCAM	*13*	Bisync communication support
*	INCLUDE BSCX21	*27*	Bisync communication support for X.21
Sensor input/output—must be grouped together in any partition			
*	PART 1		
*	INCLUDE SBCOM	*15*	Basic sensor I/O support
*	INCLUDE SBAI	*3*	Analog input support
*	INCLUDE SBAO	*3*	Analog output support
*	INCLUDE SBDIDO	*3*	Digital input/output support
*	INCLUDE SBPI	*3*	Process interrupt support
System initialization—must be in partition 1			
*	INCLUDE $PROG1	*22*	User module included in nucleus gen
*	INCLUDE IO1024	*21*	1024 IPL support
System support -- initialization—must be in partition 1			
	INCLUDE EDXINIT	*24*	Supervisor initialization
*	INCLUDE RW4963ID	*3*	4963 fixed head refresh support
*	INCLUDE INITMFA	*3*	MFA attachment initialization
*	INCLUDE INITADAP	*3*	ALPA and SMIO initialization
*	INCLUDE INIT4978	*3*	4978/4979 terminal initialization
*	INCLUDE INIT4980	*3*	4980 display initialization

IBM Series/1 Event Driven Executive

Work Sheet 3—Installation and System Generation (cont.)

Column one	Supervisor object modules	Notes	Purpose of module
Insert user initialization modules here			

NOT REQUIRED

Work Sheets for the Sample System *(continued)*

IBM Series/1 Event Driven Executive

Work Sheet 3—Installation and System Generation (cont.)

Column one	Supervisor object modules	Notes	Purpose of module
Supervisor code being moved out of partition 1 must be moved to here			
	PART 8		
	INCLUDE EDXTIO	*4*	BASIC TERMINAL SUPPORT
	INCLUDE FULLMSG	*5*	FULL MESSAGE SUPPORT
	INCLUDE IOS4979	*3*	4978/4979 DISPLAY SUPPORT
	INCLUDE IOS4974	*3*	4973/4974/4975 PRINTER SUPPORT
Location of supervisor → $EDXNUC2			
LINK $EDXNUCT,EDX002 REPLACE END			

Chapter 5. Generate a Tailored Operating System

This chapter provides the step-by-step procedures for generating a tailored operating system. In order to generate an operating system, you should have completed work sheet 2 and work sheet 3. Work sheet 2 contains the system definition statements that define processor storage and the devices attached to your Series/1. Work sheet 3 defines the software features (object modules) you should include in your supervisor to support the I/O devices defined.

Note: Before you generate a tailored operating system, EDX must be installed as described in Chapter 3, "Install EDX" on page IS-11.

The following is a summary of the steps you must perform to generate a tailored operating system:

Step 1 IPL the starter system from disk.

Step 2 Allocate required data sets.

Step 3 Edit $EDXDEF on volume ASMLIB to specify the system definition statements to match your hardware requirements.

Step 4 Edit $LNKCNTL on volume ASMLIB to specify which object modules are to be included in your supervisor.

Step 5 Edit $JOBUTIL procedure file ($SUPPREP on volume ASMLIB) to use the data sets allocated in Step 2.

Step 6 Execute $SUPPREP. Use the $JOBUTIL utility and the job stream procedure created in Step 5 to:

- Compile the system definition statements created in Step 3.
- Link edit the resulting object module, using the link edit control data set created in Step 4.
- Store your tailored operating system in a data set named $EDXNUCx on EDX002.

Step 7 Use your tailored operating system.

Step 8 Verify the system generation process (optional).

To perform these steps, you use the following utilities:

$DISKUT1	$DISKUT1 is used to allocate the data sets required in Step 2.
$FSEDIT	$FSEDIT is used to edit various data sets and to enter definition statements.
$INITDSK	$INITDSK is used to initialize devices and tailored supervisors and to allocate and initialize volumes.
$JOBUTIL	$JOBUTIL is used to execute the procedure data set.
$EDXASM	$EDXASM is used to compile the supervisor definition data set (invoked by the procedure data set).
$XPSLINK	$XPSLINK is used to link edit and format the supervisor (invoked by the procedure data set).

Step 1 - IPL the Starter System from Disk

The first step in generating your tailored operating system is to IPL the starter system you previously installed on disk. To IPL the starter system installed in Chapter 3, "Install EDX" on page IS-11, press the LOAD button on the system console.

Once your system is up and running, you can begin generating your tailored operating system.

Step 2 - Allocate Required Data Sets

The system generation process requires the use of several system utility/program preparation programs. These programs require data sets for use as work areas or input/output data sets. You must allocate the four data sets on volume EDX002 before you can generate a tailored operating system.

Note: In Chapter 3, "Install EDX" on page IS-11, it is suggested in step 7 that you assemble the sample program CALCSRC to verify starter supervisor installation. If you performed that step, EDITWORK, ASMOBJ, ASMWORK, and LINKWORK have already been allocated and need not be allocated here. Proceed to step 3.

The recommended size and required organization type of each data set is shown below.

Data set name	Size (number of records)	Organization type
EDITWORK	200	D (data)
ASMWORK	500	D (data)
ASMOBJ	300	D (data)
LINKWORK	600	D (data)

Figure 6. Required data sets for system generation

Notes:

1. The size of ASMOBJ depends on the size of the supervisor being generated. To determine the size (in records) of ASMOBJ, divide the estimated supervisor size (in bytes) by 256 to get the number of records required.

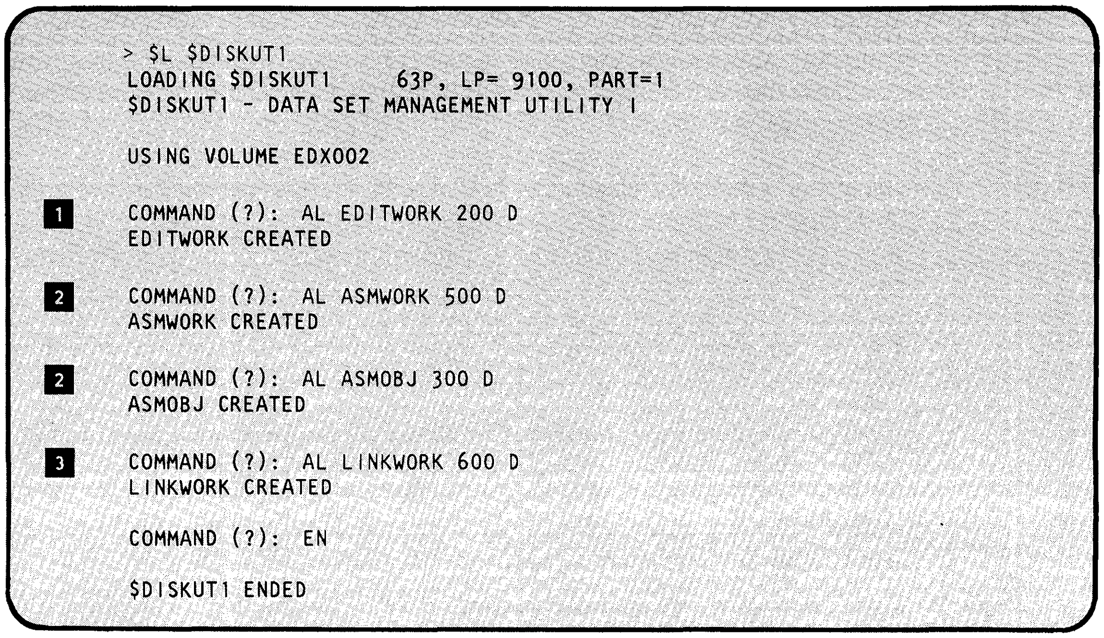
2. LINKWORK must be at least 600 records. If an end of file (EOF) occurs during the link step, the data set is too small. You then must delete and reallocate the data set specifying a larger size.

3. See $DISKUT1 in *Operator Commands and Utilities Reference* or *Operation Guide* for an explanation of organization types and instructions for allocating data sets.

To allocate the data sets, press the attention key and load $DISKUT1. Respond to the $DISKUT1 prompts as shown.

Generate a Tailored Operating System

Step 2 - Allocate Required Data Sets *(continued)*



```
  > $L $DISKUT1
  LOADING $DISKUT1     63P, LP= 9100, PART=1
  $DISKUT1 - DATA SET MANAGEMENT UTILITY I

  USING VOLUME EDX002

1 COMMAND (?):  AL EDITWORK 200 D
  EDITWORK CREATED

2 COMMAND (?):  AL ASMWORK 500 D
  ASMWORK CREATED

2 COMMAND (?):  AL ASMOBJ 300 D
  ASMOBJ CREATED

3 COMMAND (?):  AL LINKWORK 600 D
  LINKWORK CREATED

  COMMAND (?):  EN

  $DISKUT1 ENDED
```

1 EDITWORK is the name of a work data set that is required by the $FSEDIT text editing utility.

2 These data sets are used by the $EDXASM compiler. ASMOBJ is the data set in which the object module output of the compiler will be stored. ASMWORK is an assembler work data set.

3 LINKWORK is the $XPSLINK linkage editor work data set.

If you plan to use the $EDIT1N utility to edit $EDXDEF and $LNKCNTL, you must also allocate the following data sets on volume EDX002:

- $EDXDEFS at 20 records
- LINKCNTL at 111 records
- SUPPREPS at 15 records

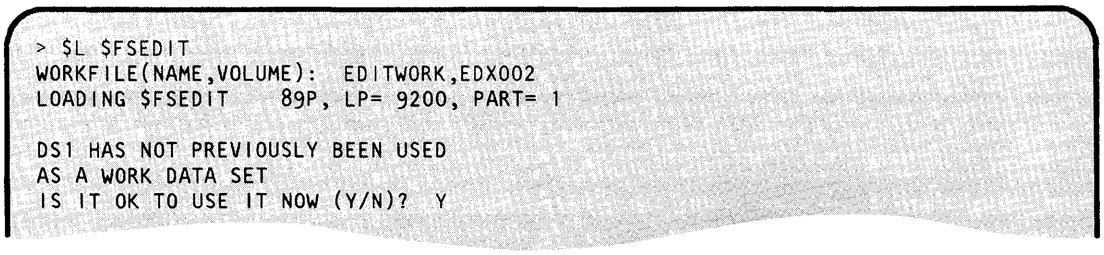
Step 3 - Edit Definition Statements to Match Hardware Configuration

Work sheet 2 contains the system definition statements that define the characteristics of processor storage and the I/O devices attached to your Series/1. During the installation procedure, a data set containing the definition statements was copied to volume EDX002 on disk. You must modify the system definition statements in this data set to match your hardware configuration.

For the starter system, this data set is named $EDXDEF.

To edit the definition statement data set:

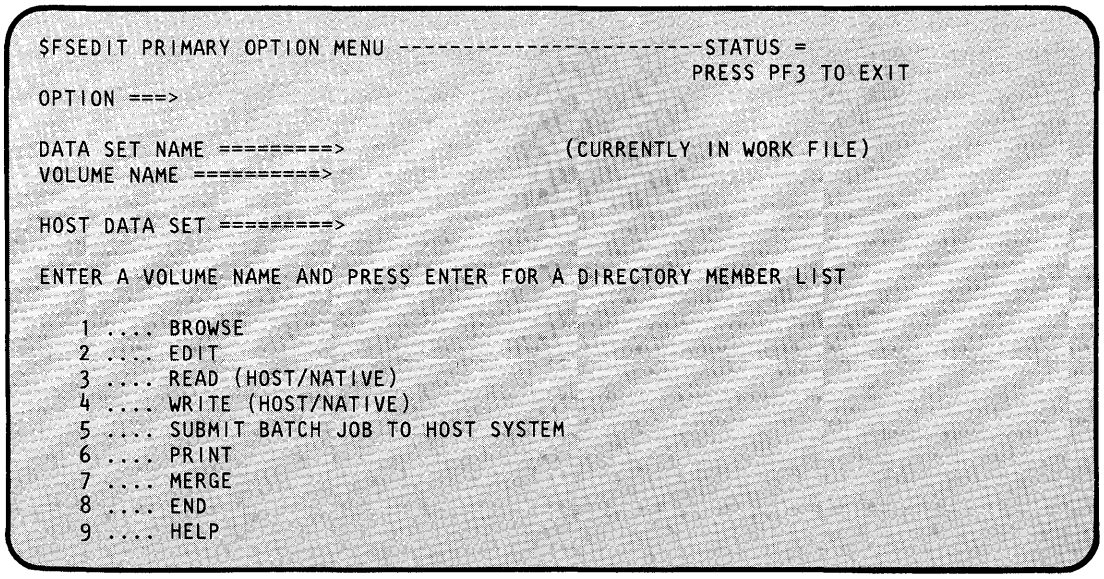
3(a) **Press the attention key and load the $FSEDIT utility.** Respond to the $FSEDIT prompts as shown.



```
> $L $FSEDIT
WORKFILE(NAME,VOLUME):  EDITWORK,EDX002
LOADING $FSEDIT    89P, LP= 9200, PART= 1

DS1 HAS NOT PREVIOUSLY BEEN USED
AS A WORK DATA SET
IS IT OK TO USE IT NOW (Y/N)?  Y
```

Note: The first time you use EDITWORK as a work file for the text editor, $FSEDIT, you are asked if you can use the EDITWORK data set as a work data set; respond YES and continue.

Once loaded, $FSEDIT displays the primary option menu.



```
$FSEDIT PRIMARY OPTION MENU -----------------------STATUS =
                                                  PRESS PF3 TO EXIT
OPTION ===>

DATA SET NAME =========>                    (CURRENTLY IN WORK FILE)
VOLUME NAME ==========>

HOST DATA SET =========>

ENTER A VOLUME NAME AND PRESS ENTER FOR A DIRECTORY MEMBER LIST

   1 .... BROWSE
   2 .... EDIT
   3 .... READ (HOST/NATIVE)
   4 .... WRITE (HOST/NATIVE)
   5 .... SUBMIT BATCH JOB TO HOST SYSTEM
   6 .... PRINT
   7 .... MERGE
   8 .... END
   9 .... HELP
```

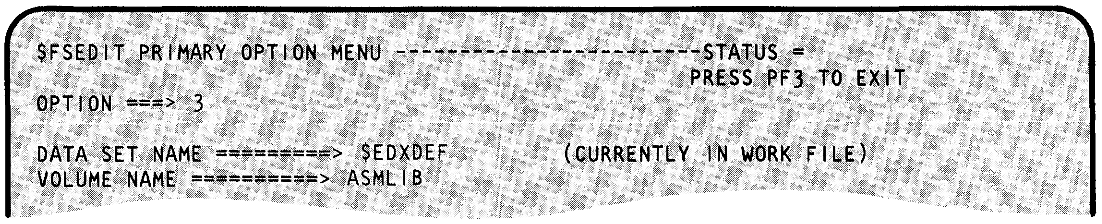
Step 3 - Edit Definition Statements to Match Hardware Configuration *(continued)*

3(b) Read a copy of the definition statement data set into the work data set.

1. Enter a **3** on the COMMAND line.
2. Enter the name of the definition statement data set on the DATA SET NAME line:

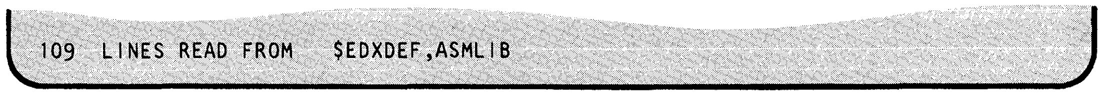
   Enter **$EDXDEF** to modify the starter system definition statements.

3. Enter EDX002 on the VOLUME NAME line.



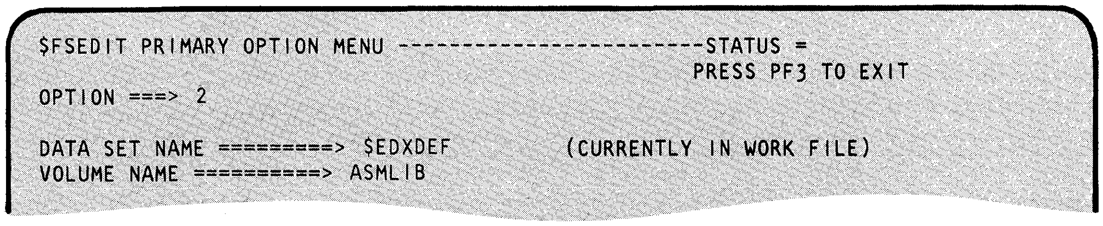
```
$FSEDIT PRIMARY OPTION MENU -----------------------STATUS =
                                                 PRESS PF3 TO EXIT
OPTION ===> 3

DATA SET NAME =========> $EDXDEF         (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> ASMLIB
```

Press the enter key. $FSEDIT reads the requested data set into the work data set and issues the following message:



```
109  LINES READ FROM   $EDXDEF,ASMLIB
```

3(c) **Select option 2 (EDIT) to edit $EDXDEF.** Enter a 2 on the OPTION line.



```
$FSEDIT PRIMARY OPTION MENU -----------------------STATUS =
                                                 PRESS PF3 TO EXIT
OPTION ===> 2

DATA SET NAME =========> $EDXDEF         (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> ASMLIB
```

Press the enter key. $FSEDIT displays the contents of the data set. Figure 7 on page IS-83 shows the contents of $EDXDEF.

Step 3 - Edit Definition Statements to Match Hardware Configuration *(continued)*

```
$EDXDEF  CSECT
         $ID
*                          XPS
*                  EVENT DRIVEN EXECUTIVE
*              VERSION 5, MODIFICATION LEVEL 0
*
* THE FOLLOWING DEFINES THE STARTER SUPERVISOR AS SHIPPED ON THE
* DISKETTE LABELED XS5001.  FOR COMPLETE DESCRIPTIONS OF THESE
* STATEMENTS OR ANY OTHER SYSTEM DEFINITION STATEMENTS, REFER TO
* THE INSTALLATION AND SYSTEM GENERATION GUIDE, SC34-0646.
*
* ONLY STATEMENTS WITHOUT AN ASTERISK (*) IN COLUMN 1 ARE ASSEMBLED.
*
*----------------------------------------------------------------------
* BASIC STORAGE DEFINITIONS
*----------------------------------------------------------------------
* PARTITION NUMBER:    1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
      SYSTEM MAXPROG=(05,05,05,05,05,05,05,05),                       C
               PARTS=(32,32,32,32,32,32,32,32)
*----------------------------------------------------------------------
* BASIC DEFINITION FOR A 4969 TAPE
*----------------------------------------------------------------------
*        TAPE   DEVICE=4969,ADDRESS=4C,ID=TAPE01,                     C
*               DENSITY=1600,LABEL=SL,TASK=YES
*----------------------------------------------------------------------
* BASIC DEFINITION FOR A 4962-1 OR -2 DISK        (9.3 MB)
*----------------------------------------------------------------------
*        DISK   DEVICE=4962-1F,ADDRESS=03
*----------------------------------------------------------------------
* BASIC DEFINITION FOR A 4963-23 DISK              (23 MB)
*----------------------------------------------------------------------
         DISK   DEVICE=4963-23,ADDRESS=48
*----------------------------------------------------------------------
* BASIC DEFINITION FOR A 4967-2 OR -4 DISK        (200 MB)
*----------------------------------------------------------------------
*        DISK   DEVICE=4967-2,ADDRESS=48
*----------------------------------------------------------------------
* BASIC DEFINITION FOR A DDSK-30 DISK              (30 MB)
*----------------------------------------------------------------------
*        DISK   DEVICE=DDSK-30,ADDRESS=44
*----------------------------------------------------------------------
* DISKETTE DEFINITIONS
*----------------------------------------------------------------------
         DISK   DEVICE=4964,ADDRESS=02
         DISK   DEVICE=4965,ADDRESS=44
         DISK   DEVICE=4965,ADDRESS=45
         DISK   DEVICE=4966,ADDRESS=22,END=YES
```

Figure 7 (Part 1 of 3). $EDXDEF data set

Step 3 - Edit Definition Statements to Match Hardware Configuration *(continued)*

```
*-------------------------------------------------------------------
* TIMER DEFINITION
*-------------------------------------------------------------------
*        TIMER  ADDRESS=40
*-------------------------------------------------------------------
* ADAPTER DEFINITIONS
*-------------------------------------------------------------------
SMI004   ADAPTER   TYPE=SMIO,ADDRESS=80,                            C
               DEVICES=($SYSLOG)
MFA58    ADAPTER   TYPE=MFA,ADDRESS=58,                             C
               DEVICES=($SYSLOGB,MPRTR1,MPRTR2),END=YES
*ALPACO  ADAPTER   TYPE=ALPA,ADDRESS=CO,                            C
*              DEVICES=(MPRTR3),END=YES
*-------------------------------------------------------------------
* VIRTUAL TERMINAL DEFINITIONS
*-------------------------------------------------------------------
*CDRVTA   TERMINAL  DEVICE=VIRT,ADDRESS=CDRVTB,SYNC=YES
*CDRVTB   TERMINAL  DEVICE=VIRT,ADDRESS=CDRVTA,SYNC=NO
*-------------------------------------------------------------------
* SERIES/1 TO SERIES/1 TERMINAL DEFINITION
*-------------------------------------------------------------------
*S1S1     TERMINAL  DEVICE=S1S1,ADDRESS=25
*-------------------------------------------------------------------
* GENERAL PURPOSE INTERFACE BUS TERMINAL DEFINITION
*-------------------------------------------------------------------
*GPIB     TERMINAL  DEVICE=GPIB,ADDRESS=25
*-------------------------------------------------------------------
* TERMINAL DEFINITIONS
*-------------------------------------------------------------------
*$SYSLOG  TERMINAL  DEVICE=4979,ADDRESS=04,HDCOPY=$SYSPRTR,PART=2
*$SYSLOG  TERMINAL  DEVICE=4978,ADDRESS=24,HDCOPY=$SYSPRTR,PART=2
$SYSLOG  TERMINAL  DEVICE=4980,ADDRESS=80,HDCOPY=$SYSPRTR,PART=2,   C
               PORT=0,SECADDR=AA,ADAPTER=SMIO
$SYSLOGA TERMINAL  DEVICE=TTY,ADDRESS=00,CRDELAY=4,PAGSIZE=24,      C
               BOTM=23,SCREEN=YES,PART=2
$SYSLOGB TERMINAL  DEVICE=ACCA,ADDRESS=59,MODE=3101B,               C
               ADAPTER=MFA,LMODE=RS422,PART=2
$SYSPRTR TERMINAL  DEVICE=4974,ADDRESS=01
MPRTR1   TERMINAL  DEVICE=4975-01L,ADDRESS=5A,ADAPTER=MFA,          C
               CHARSET=USCA
MPRTR2   TERMINAL  DEVICE=4975-02L,ADDRESS=5B,ADAPTER=MFA,          C
               CHARSET=USCA,END=YES
*REMSUPT TERMINAL  DEVICE=ACCA,ADDRESS=08,BITRATE=1200,             C
*              LMODE=SWITCHED,ADAPTER=SINGLE,PF1=D886,              C
*              CODTYPE=EBASC,ATTN=D816,PAGSIZE=24,SCREEN=YES
```

Figure 7 (Part 2 of 3). $EDXDEF data set

Step 3 - Edit Definition Statements to Match Hardware Configuration *(continued)*

```
*MPRTR3  TERMINAL  DEVICE=5219,ADDRESS=C0,ADAPTER=ALPA,                 C
*              PORT=0,SECADDR=00
*MPRTR3  TERMINAL  DEVICE=5224,ADDRESS=C0,ADAPTER=ALPA,                 C
*              PORT=0,SECADDR=00
*MPRTR3  TERMINAL  DEVICE=5225,ADDRESS=C0,ADAPTER=ALPA,                 C
*              PORT=0,SECADDR=00,END=YES
*---------------------------------------------------------------------
* COMMUNICATION LINE DEFINITION
*---------------------------------------------------------------------
*         BSCLINE ADDRESS=09,END=YES
*---------------------------------------------------------------------
* SYSTEM  COMMON
*---------------------------------------------------------------------
$SYSCOM  CSECT
         QCB
         QCB
         ECB
         ECB
         ENTRY     $EDXPTCH
$EDXPTCH DATA      128F'0'             SYSTEM PATCH AREA
         END
```

Figure 7 (Part 3 of 3). $EDXDEF data set

3(d) Edit $EDXDEF

Use $FSEDIT to add to, modify, or delete from the contents of $EDXDEF to match the set of system definition statements you prepared when filling out work sheet 2. For information on editing a data set, see $FSEDIT in the *Operator Commands and Utilities Reference*.

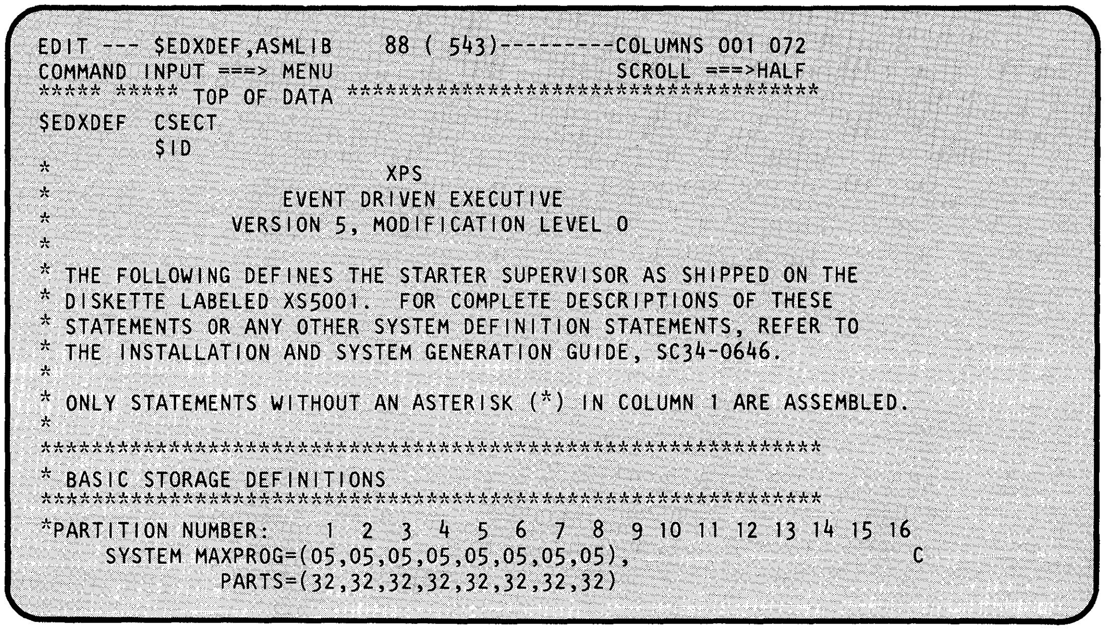
When editing this data set, be sure that:

- Continuation indicators in column 72 are not removed.
- If required, a continuation character is placed in column 72 and the statement is continued starting in column 16 of the next line.
- A system definition statement does not extend beyond column 72.

After you modify the definition data set to match the system definition statements specified the work sheet, you need to save the changes.

Step 3 - Edit Definition Statements to Match Hardware Configuration *(continued)*

3(e) **Enter MENU on the COMMAND INPUT line to end the EDIT mode as shown.**



```
EDIT --- $EDXDEF,ASMLIB    88 ( 543)---------COLUMNS 001 072
COMMAND INPUT ===> MENU                      SCROLL ===>HALF
***** ***** TOP OF DATA **************************************
$EDXDEF  CSECT
         $ID
*                      XPS
*                 EVENT DRIVEN EXECUTIVE
*              VERSION 5, MODIFICATION LEVEL 0
*
* THE FOLLOWING DEFINES THE STARTER SUPERVISOR AS SHIPPED ON THE
* DISKETTE LABELED XS5001.  FOR COMPLETE DESCRIPTIONS OF THESE
* STATEMENTS OR ANY OTHER SYSTEM DEFINITION STATEMENTS, REFER TO
* THE INSTALLATION AND SYSTEM GENERATION GUIDE, SC34-0646.
*
* ONLY STATEMENTS WITHOUT AN ASTERISK (*) IN COLUMN 1 ARE ASSEMBLED.
*
**************************************************************
* BASIC STORAGE DEFINITIONS
**************************************************************
*PARTITION NUMBER:     1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
     SYSTEM MAXPROG=(05,05,05,05,05,05,05,05),                      C
            PARTS=(32,32,32,32,32,32,32,32)
```

Press the enter key. $FSEDIT returns to the primary option menu.

Step 3 - Edit Definition Statements to Match Hardware Configuration *(continued)*

3(f) **Save the changed data set.**

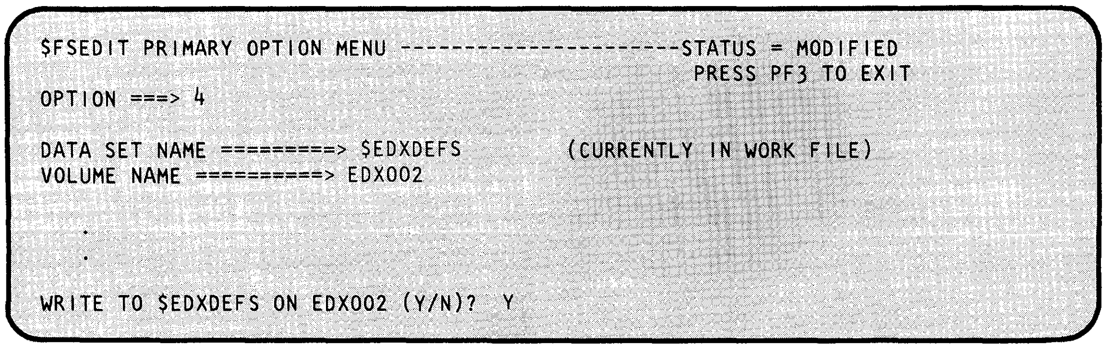
You may want to maintain more than one supervisor. For example, you may need one system for development and one for production. It is recommended that you save the definition statements that are unique for each supervisor in a separate data set. The name assigned to the data set should be named $EDXDEFx, where x is any alphanumeric character.

1. Enter a **4** on the OPTION line.

2. Enter the data set name and volume name as shown.

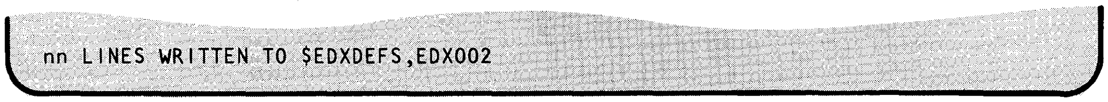
3. Press the enter key.

4. Enter a **Y** in response to the verification prompt and press the enter key.



```
$FSEDIT PRIMARY OPTION MENU ---------------------STATUS = MODIFIED
                                                  PRESS PF3 TO EXIT
OPTION ===> 4

DATA SET NAME =========> $EDXDEFS        (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> EDX002

   .
   .

WRITE TO $EDXDEFS ON EDX002 (Y/N)?  Y
```

$FSEDIT saves the contents of the work data set in the data set specified and issues the following message:



```
nn LINES WRITTEN TO $EDXDEFS,EDX002
```

where nn indicates the number of lines in the data set.

Step 3 - Edit Definition Statements to Match Hardware Configuration *(continued)*

Example: Here is an example of the definition data set as modified for the sample system.

```
$EDXDEF  CSECT
         $ID
*                              XPS
*                    EVENT DRIVEN EXECUTIVE
*                VERSION 5, MODIFICATION LEVEL 0
*
* THE FOLLOWING DEFINES THE STARTER SUPERVISOR AS SHIPPED ON THE
* DISKETTE LABELED XS5001.  FOR COMPLETE DESCRIPTIONS OF THESE
* STATEMENTS OR ANY OTHER SYSTEM DEFINITION STATEMENTS, REFER TO
* THE INSTALLATION AND SYSTEM GENERATION GUIDE, SC34-0646.
*
* ONLY STATEMENTS WITHOUT AN ASTERISK (*) IN COLUMN 1 ARE ASSEMBLED.
*
*---------------------------------------------------------------
* BASIC STORAGE DEFINITIONS
*---------------------------------------------------------------
         SYSTEM MAXPROG=(1,10,10,5,5),PARTS=(0,24,16,20,32),   C
                COMMON=(0,5,0,2,3)
*---------------------------------------------------------------
* BASIC DEFINITION FOR A 4962-1 OR -2 DISK     (9.3 MB)
*---------------------------------------------------------------
         DISK   DEVICE=4962-1F,ADDRESS=03,TASK=YES
*---------------------------------------------------------------
* DISKETTE DEFINITIONS
*---------------------------------------------------------------
         DISK    DEVICE=4964,ADDRESS=02,TASK=YES,END=YES
*---------------------------------------------------------------
* TERMINAL DEFINITIONS
*---------------------------------------------------------------
$SYSPRTR TERMINAL DEVICE=4973,ADDRESS=21
$SYSLOG  TERMINAL DEVICE=4979,ADDRESS=04,HDCOPY=$SYSPRTR
$SYSLOGA TERMINAL DEVICE=4979,ADDRESS=24,HDCOPY=$SYSPRTR,END=YES
*---------------------------------------------------------------
* SYSTEM COMMON
*---------------------------------------------------------------
$SYSCOM  CSECT
*
* NONE DEFINED
*
          ENTRY     $EDXPTCH
$EDXPTCH  DATA      128F'0'        SYSTEM PATCH AREA
          END
```

Figure 8. Edited $EDXDEFx Data Set

Step 4 - Edit Link Control Data Set to Include Software Support

The link-control data set contains all the supervisor object modules needed to generate an operating system. However, the statements for the modules that are not required for the starter system are preceded by an asterisk in column one. The asterisk causes the linkage editor to treat the statement as a comment rather than a control statement. You must edit the link-control data set to remove or add asterisks, as appropriate, to match your system requirements.

The link-control data set is on the volume named ASMLIB. For the starter system it is named $LNKCNTL.

To edit the link-control data set:

4(a) Read the link-control data set into the work data set EDITWORK.

1. Enter a **3** on the OPTION line.
2. Enter **$LNKCNTL** on the DATA SET NAME line.
3. Enter **ASMLIB** on the VOLUME NAME line.
4. Press the enter key.

```
$FSEDIT PRIMARY OPTION MENU ------------------------STATUS =
                                                     PRESS PF3 TO EXIT
OPTION ===> 3

DATA SET NAME =========> $LNKCNTL        (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> ASMLIB
```

$FSEDIT reads the requested data set into the EDITWORK data set and issues the following message:

```
236   LINES READ FROM   $LNKCNTL,ASMLIB
```

4(b) **Select option 2 (EDIT) to edit $LNKCNTL.** Enter a **2** on the OPTION line.

```
$FSEDIT PRIMARY OPTION MENU ------------------------STATUS =
                                                     PRESS PF3 TO EXIT
OPTION ===> 2

DATA SET NAME =========> $LNKCNTL       (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> ASMLIB
```

Press the enter key. $FSEDIT displays the $LNKCNTL data set. Figure 9 on page IS-90 shows the contents of $LNKCNTL.

Step 4 - Edit Link Control Data Set to Include Software Support *(continued)*

```
*                          STARTER SYSTEM
*                       EVENT DRIVEN EXECUTIVE
*                  XPS - VERSION 5, MODIFICATION LEVEL 0
*
*        COMMENTS MAY BE INCLUDED BY AN '*' IN COLUMN 1
*        USE THIS TECHNIQUE TO OMIT UNNEEDED MODULES
*
*OPTION NOVERLAY         *25*  NO OVERLAY STRUCTURE
*-------------------------------------------------------------------
*  SUPERVISOR SUPPORT    - MUST BE FIRST AND IN PARTITION 1
*-------------------------------------------------------------------
 PART 1
 VOLUME  XS5002                 DEFAULT VOLUME FOR INCLUDE MODULES
*OVLAREA OVLSTART OVLEND *23*   USER DEFINED OVERLAY AREA
 INCLUDE EDXSYS           *1*   SYSTEM TABLES AND WORK AREAS
 INCLUDE ASMOBJ,EDX002    *1*   OUTPUT FROM USER SYSTEM GENERATION
 INCLUDE EDXSVCX          *1*   TASK SUPERVISOR
*INCLUDE $DBUGNUC         *2*   RESIDENT $DEBUG SUPPORT
 INCLUDE EDXALU           *1*   EDL INSTRUCTION EMULATOR
 INCLUDE EDXSTART         *1*   INITIALIZATION & ERROR HANDLER
*INCLUDE SWAITM           *3*   WAIT ON MULTIPLE EVENTS
*INCLUDE SUPVIO           *28*  MAKE SUPERVISOR AND COMMON AREA STATIC
*
*-------------------------------------------------------------------
*  TIMER SUPPORT         - MUST BE IN PARTITION 1
*-------------------------------------------------------------------
*INCLUDE EDXTIMER         *12*  4955 TIMER SUPPORT (7840)
*INCLUDE EDXTIMR2         *12*  4952/4954/4956 TIMER SUPPORT
*
*-------------------------------------------------------------------
*  EXIO SUPPORT          - MUST BE IN PARTITION 1
*-------------------------------------------------------------------
*INCLUDE IOSEXIO           *3*   EXIO DEVICE CONTROL SUPPORT
*INCLUDE EXIOTRC           *3*   EXIO TRACE OPTION
*
*-------------------------------------------------------------------
*  ERROR LOGGING         - MUST BE IN PARTITION 1
*-------------------------------------------------------------------
*INCLUDE CIRCBUFF         *16*  PROGRAM/MACHINE CHECK LOGGING
*
*-------------------------------------------------------------------
*  OPTIONAL FUNCTION SUPPORT - MUST BE IN PARTITION 1
*-------------------------------------------------------------------
 INCLUDE RLOADER          *17*  RELOCATING PROGRAM LOADER
*INCLUDE STORMGR          *3*   UNMAPPED STORAGE MANAGER SUPPORT
*INCLUDE IAMQCB           *20*  IAM QCB NEEDED FOR IAM SUPPORT
 INCLUDE PWRAM80          *26*  4980 POWER ON RAM
*
```

Figure 9 (Part 1 of 5). $LNKCNTL data set contents

```
*------------------------------------------------------------------
*  HOST COMMUNICATIONS SUPPORT - MUST BE IN PARTITION 1
*------------------------------------------------------------------
*INCLUDE TPCOM          *14*  HOST COMMUNICATION SUPPORT
*
*------------------------------------------------------------------
*  TRANSLATION TABLES    - MUST BE IN PARTITION 1
*------------------------------------------------------------------
 INCLUDE TRASCII        *10*  1310,2095/2096,7850 ACCA/TTY TRANSLATION
 INCLUDE TREBASC        *10*  1610,2091/2092 ACCA TRANSLATION
*INCLUDE TREBCD         *11*  2741,PROC EBDC TRANSLATION
*INCLUDE TRCRSP         *11*  2741 CORRESPONDENCE TRANSLATION
*
*------------------------------------------------------------------
*  EXTENDED ADDRESS MODE - MAY BE INCLUDED IN PARTITION 1 TO 8
*------------------------------------------------------------------
*INCLUDE SRMGR          *29*   INCLUDE FOR EXTENDED ADDRESS MODE SUPPORT
*
*------------------------------------------------------------------
*  DISK(ETTE) SUPPORT    - MUST BE GROUPED TOGETHER
*                        - MAY BE INCLUDED IN PARTITION 1 TO 8
*------------------------------------------------------------------
 PART1
 INCLUDE DISKIO           *3*   BASIC DISK(ETTE) SUPPORT
 INCLUDE D49624           *3*   4962/4964 DISK(ETTE) SUPPORT
 INCLUDE D4963A           *3*   4963/4967/DDSK-30 DISK SUPPORT
 INCLUDE D4966A           *3*   4965/4966 DISKETTE SUPPORT
*INCLUDE D1024          *3,21* 1024 BYTES/SECTOR DISKETTE SUPPORT
*INCLUDE D4969A           *3*   BASIC TAPE SUPPORT
*
*------------------------------------------------------------------
*  TERMINALS             - MUST BE GROUPED TOGETHER
*                        - MAY BE INCLUDED IN PARTITION 1 TO 8
*------------------------------------------------------------------
 PART 1
 INCLUDE EDXTIO           *4*   BASIC TERMINAL SUPPORT
 INCLUDE MINMSG           *5*   MINIMUM MESSAGE SUPPORT
*INCLUDE FULLMSG          *5*   FULL MESSAGE SUPPORT
 INCLUDE IOS3101          *7*   ACCA 3101B SUPPORT
 INCLUDE IOS4979          *3*   4978/4979/4980 DISPLAY SUPPORT
 INCLUDE IOS4974          *3*   4973/4974/4975/5219/5224/5225 SUPPORT
*INCLUDE IOS4975A         *3*   4975-01A PRINTER SUPPORT
 INCLUDE IOSACCA          *4*   ACCA DEVICE HANDLER
*INCLUDE ACCATRC          *3*   ACCA TRACE OPTION
 INCLUDE IOSTERM        *A,6*   ACCA/TTY/2741/4013 SUPPORT
 INCLUDE IOSTTY           *3*   ASR 33/35/3101/3101C TTY SUPPORT
*INCLUDE IOS2741          *3*   2741 TERMINAL SUPPORT
*INCLUDE IOS4013          *3*   DIGITAL I/O TERMINAL SUPPORT
*INCLUDE IOSGPIB          *3*   GPIB SUPPORT
*INCLUDE IOSS1S1          *3*   SERIES/1 - SERIES/1 SUPPORT
*INCLUDE IOSVIRT        *3,9*   VIRTUAL TERMINAL SUPPORT
*INCLUDE IOSPOOL          *3*   SPOOLING SUPPORT
*INCLUDE EBFLCVT         *18*   EBCDIC/FLOATING POINT CONV.
*
```

Figure 9 (Part 2 of 5). $LNKCNTL data set contents

Step 4 - Edit Link Control Data Set to Include Software Support *(continued)*

```
*---------------------------------------------------------------------
*  FLOATING POINT        - MAY BE INCLUDED IN PARTITION 1 TO 8
*---------------------------------------------------------------------
*PART 1
*INCLUDE EDXFLOAT         *3*   FLOATING POINT ARITHMETIC
*
*---------------------------------------------------------------------
*  QUEUE I/O             - MAY BE INCLUDED IN PARTITION 1 TO 8
*---------------------------------------------------------------------
*PART 1
*INCLUDE QUEUEIO         *19*   QUEUE PROCESSING SUPPORT
*
*---------------------------------------------------------------------
*  BISYNC COMMUNICATIONS - MUST BE GROUPED TOGETHER
*                        - MAY BE INCLUDED IN PARTITION 1 TO 8
*---------------------------------------------------------------------
*PART 1
*INCLUDE BSCAM           *13*   BISYNC COMMUNICATION SUPPORT
*INCLUDE BSCX21          *27*   BISYNC COMMUNICATION SUPPORT FOR X.21
*
*---------------------------------------------------------------------
*  SENSOR INPUT/OUTPUT   - MUST BE GROUPED TOGETHER
*                        - MAY BE INCLUDED IN PARTITION 1 TO 8
*---------------------------------------------------------------------
*PART 1
*INCLUDE SBCOM           *15*   BASIC SENSOR I/O SUPPORT
*INCLUDE SBAI             *3*   ANALOG INPUT SUPPORT
*INCLUDE SBAO             *3*   ANALOG OUTPUT SUPPORT
*INCLUDE SBDIDO           *3*   DIGITAL INPUT/OUTPUT SUPPORT
*INCLUDE SBPI             *3*   PROCESS INTERRUPT SUPPORT
*
*---------------------------------------------------------------------
*  SYSTEM INITIALIZATION - MUST BE IN PARTITION 1
*---------------------------------------------------------------------
*INCLUDE $PROG1          *22*   USER MODULE INCLUDED IN NUCLEUS GEN
*INCLUDE IO1024          *21*   1024 IPL SUPPORT
*
*---------------------------------------------------------------------
*  SYSTEM SUPPORT -- INITIALIZATION - MUST BE IN PARTITION 1
*---------------------------------------------------------------------
 INCLUDE EDXINIT         *24*   SUPERVISOR INITIALIZATION
*
```

Figure 9 (Part 3 of 5). $LNKCNTL data set contents

Step 4 - Edit Link Control Data Set to Include Software Support *(continued)*

```
*---------------------------------------------------------------
*  INSERT USER INITIALIZATION MODULES HERE
*---------------------------------------------------------------
*
*---------------------------------------------------------------
*  SUPERVISOR CODE BEING MOVED OUT OF
*  PARTITION 1 MUST BE MOVED TO HERE
*---------------------------------------------------------------
*PART 2
*INCLUDE SUPVIO         *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 3
*INCLUDE SUPVIO         *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 4
*INCLUDE SUPVIO         *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 5
*INCLUDE SUPVIO         *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 6
*INCLUDE SUPVIO         *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 7
*INCLUDE SUPVIO         *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 8
*INCLUDE SUPVIO         *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*-------------------------------------------------------
LINK $EDXNUCT,EDX002 REPLACE END
*-------------------------------------------------------
*
*-------------------------------------------------------
*  PROGRAMMING NOTES
*-------------------------------------------------------
*1*  MUST BE INCLUDED FIRST AND IN THIS ORDER
*2*  REQUIRED FOR PROGRAM DEBUGGING ($DEBUG)
*3*  OPTIONAL MODULE;  REQUIRED IF THE DEVICE OR FEATURE IS USED
*4*  REQUIRED IF ANY TERMINALS ARE INSTALLED, INCLUDING 4973,4974,4975,
*    5219, 5224, 5225, 4978, 4979, 4980, GPIB, S/1-S/1, ACCA, TTY,
*    2741, 3101, 4013, ETC.
*5*  MUTUALLY EXCLUSIVE MODULES; ONE, BUT NOT BOTH, IS REQUIRED.
*6*  IOSTERM IS REQUIRED IF IOSTTY, IOS2741 AND/OR IOS4013 ARE INCLUDED.
*    IOSTERM MUST ALSO BE INCLUDED WHERE USING NON-3101B ACCA TERMINALS.
*    SEE NOTES 7, 8, 10, 11.
*7*  IOS3101 IS REQUIRED IF A 3101 TERMINAL IS TO BE USED IN BLOCK MODE
*    (DEVICE=ACCA,MODE=3101B). IOSACCA MUST ALSO BE INCLUDED.
*    SEE NOTES 6, 8, 10, 11.
*8*  IOSACCA IS REQUIRED IF DEVICE=ACCA WAS SPECIFIED IN THE SYSGEN.
*    APPROPRIATE TRANSLATION TABLE(S) MUST BE INCLUDED.
*    SEE NOTES 6, 7, 10, 11.
*9*  REQUIRED IF USING REMOTE MANAGEMENT UTILITY WITH PASSTHRU FUNCTION
*10* TRASCII AND/OR TREBASC ARE REQUIRED IF AN ACCA OR TTY TYPE ATTACH-
*    MENT CARD IS USED. FEATURE CARDS #1310, #2095/2096 AND #7850
*    REQUIRE TRASCII TRANSLATION TABLES. #1610 AND #2091/2092 REQUIRE
*    TREBASC TRANSLATION TABLES.
*11* TREBCD AND/OR TRCRSP  ARE REQUIRED IF IOS2741 IS INCLUDED. 2741
*    TERMINALS MAY USE EITHER TREBCD OR TRCRSP.
*    DEVICE=PROC NORMALLY USES TREBCD.
```

Figure 9 (Part 4 of 5). $LNKCNTL data set contents

Step 4 - Edit Link Control Data Set to Include Software Support *(continued)*

```
*12* ATTACHED TIMERS (FEATURE 7840) AND THE 4952/4954/4956 NATIVE TIMER
*    ARE MUTUALLY EXCLUSIVE.  SELECT THE TIMER SUPPORT REQUIRED FOR
*    YOUR CONFIGURATION OR NONE IF NO TIMER SUPPORT IS REQUIRED.
*13* REQUIRED FOR BINARY SYNCHRONOUS COMMUNICATION USING BSCREAD/
*    BSCWRITE OR REMOTE MANAGEMENT UTILITY SUPPORT.
*14* REQUIRED FOR COMMUNICATION TO A S/370 WITH THE EDX HOST
*    COMMUNICATION FACILITY
*15* REQUIRED IF ANY SENSOR I/O SUPPORT IS TO BE USED
*    (AI,AO,DI,DO, OR PI)
*16* REQUIRED IF THE IN STORAGE PROGRAM CHECK/MACHINE CHECK LOG
*    IS TO BE KEPT
*17* REQUIRED IF PROGRAMS ARE TO BE LOADED FROM DISK(ETTE).
*    IF NOT INCLUDED, AN APPLICATION PROGRAM MUST BE LINK EDITED
*    WITH THE SUPERVISOR.
*18* REQUIRED FOR DATA FORMATTING OPERATIONS (GETEDIT, PUTEDIT, FORMAT)
*19* REQUIRED FOR QUEUEING OPERATIONS (FIRSTQ, NEXTQ, LASTQ, DEFINEQ)
*20* ONLY NEEDED IF IAM (5719-AM4) IS EVER TO BE INSTALLED.
*21* OPTIONAL MODULE;  REQUIRED IF THE SYSTEM IS TO BE IPLED
*    FROM A 1024 BYTE/SECTOR DISKETTE. INCLUDE BEFORE EDXINIT.
*22* USER PROGRAM $PROG1, LINKED WITH THE SUPERVISOR, MUST
*    PRECEED EDXINIT.  THE VOLUME NAME SHOULD REFLECT THE VOLUME
*    WHERE THIS OBJECT MODULE RESIDES.
*23* OPTIONAL LINK CONTROL STATEMENT TO DEFINE THE OVERLAY AREA
*    DISKBUFR IS EQUATED TO THE START OF SEGINIT, THE START OF THE
*    OVERLAY AREA (IN THIS CASE, OVLSTART) IS EQUATED TO DISKBUFR+512,
*    AND THE END OF THE OVERLAY AREA IS EQUATED TO THE END OF DISKINIT
*    (OVLEND).  THE USER MAY DEFINE ANY ENTRIES FOR THE START AND THE
*    END OF THE OVERLAY AREA, BUT THE SIZE MUST BE LARGER THAN THE
*    LARGEST OVERLAY INCLUDED OR USER WILL GET A LINK ERROR.
*24* REQUIRED, AND MUST FOLLOW ALL OF THE PREVIOUSLY LISTED MODULES.
*    ALL OTHER INITIALIZATION MODULES MUST FOLLOW EDXINIT.  AFTER
*    INITIALIZATION IS COMPLETE, ALL STORAGE AFTER EDXINIT IS GIVEN
*    BACK TO THE USER AT THE FIRST PAGE BOUNDARY.
*25* INCLUDE ONLY IF USER WISHES TO SYSGEN SUPERVISOR IN A NONOVERLAY
*    INITIALIZATION STRUCTURE (OR TAILORED INITIALIZATION STRUCTURE).
*    IF THIS OPTION IS CHOSEN, THEN IT IS THE USER'S RESPONSIBILITY
*    TO INCLUDE MODULES THAT CANNOT BE INCLUDED BY THE PREPROCESSOR
*    (FOR EXAMPLE, RW4963ID, INITMFA, INITADAP, INIT4978, INIT4980).
*26* INCLUDE ONLY IF USER WISHES TO RE-RAM TERMINALS, IN CASE OF
*    POWER ON/OFF, WITHOUT RE-IPLING THE SYSTEM.
*27* REQUIRED FOR X.21 SWITCHED NETWORK SUPPORT WITH BISYNC
*    COMMUNICATION SUPPORT.
*28* MAKES ALL THE COMMON AND SUPERVISOR AREA OF EACH PARTITION THAT
*    SUPVIO IS INCLUDED IN STATIC (ONLY VALID FOR EXTENDED ADDRESS
*    MODE SUPPORT).
*29* INCLUDE FOR EXTENDED ADDRESS MODE SUPPORT.  IF INCLUDED, RLOADER
*    WILL AUTOMATICALLY BE INCLUDED; EDXTIMR2 WILL ALSO BE INCLUDED
*    AUTOMATICALLY IF EDXTIMER IS NOT INCLUDED.  FOR MORE INFORMATION
*    REFER TO THE 'IBM SERIES/1 EVENT DRIVEN EXECUTIVE EXTENDED ADDRESS
*    MODE AND PERFORMANCE ANALYZER USER GUIDE', SC34-0591.
*A*  WILL BE INCLUDED AUTOMATICALLY ONLY IF ONE OF THE FOLLOWING IS
*    INCLUDED : IOSTTY, IOS2741 AND/OR IOS4013
*    (SEE NOTE 6 TO DETERMINE IF THE USER MUST INCLUDE THIS MODULE.
*    FOR EXAMPLE, THE CASE WHERE IOS3101 IS REQUIRED IS NOT AUTOMATIC.)
```

Figure 9 (Part 5 of 5). $LNKCNTL data set contents

Note: Refer to the *EDX Extended Address Mode and Performance Analyzer User Guide*, SC34-0591, for information about Extended Address Mode support.

4(c) Edit $LNKCNTL

Work sheet 3 contains the names of the supervisor object modules you selected to provide the software support for the I/O devices attached to your Series/1. Match the link control data set to the work sheet, inserting asterisks to omit modules or removing asterisks to include the appropriate modules.

Note: Instead of deleting undesired statements, it is preferable to insert an asterisk in column one. The asterisk causes the linkage editor $XPSLINK to treat the statement as a comment rather than a control statement. In addition, by not deleting undesired statements, you have a record of the support you decided to leave out. This can be helpful if problems develop with the generated operating system.

The number on the PART statement preceding a group of modules specifies the partition where the modules are to be located. Some groups of supervisor object modules must remain located in partition 1; others can be located outside of partition 1. If you are generating a single partition supervisor, you do not have to change the PART statements as shown in the link-control data set. If you are generating a multipartition supervisor, you should have indicated the number of the partition where you want to locate a specific group of modules on work sheet 3. In editing the link-control data to match your requirements for a multipartition supervisor, you must physically move any group of modules being located outside of partition 1 to the location specified on work sheet 3 in the $LNKCNTL data set.

All groups of object modules defined to be in a specific partition must be adjacent to one another. For example, you can specify partitions 1, 2, 2, 3, 3, 4, 4 or 1, 2, 2, 4, 3, 3, 5, 5. You cannot specify partitions 1, 1, 2, 1, 3, 4, 3. This means that if you have already defined a group of modules to be in a specific partition and wish to place another group of modules in that same partition, you must physically move the second group of modules to follow PART statement defining that partition.

If you are going to include your own initialization routines, enter the module names following the area specified as INSERT USER INITIALIZATION MODULES HERE If the initialization routines are to be treated as overlay segments, be sure each module or group of modules is preceded by an OVERLAY statement. In addition, the initialization modules must be preceded by a PART statement specifying partition 1.

Be sure that the name of the supervisor you are generating is different from the name of the current supervisor.

Step 4 - Edit Link Control Data Set to Include Software Support *(continued)*

Example: Define a supervisor without default overlay structure, and two groups of object modules to be located outside of partition 1.

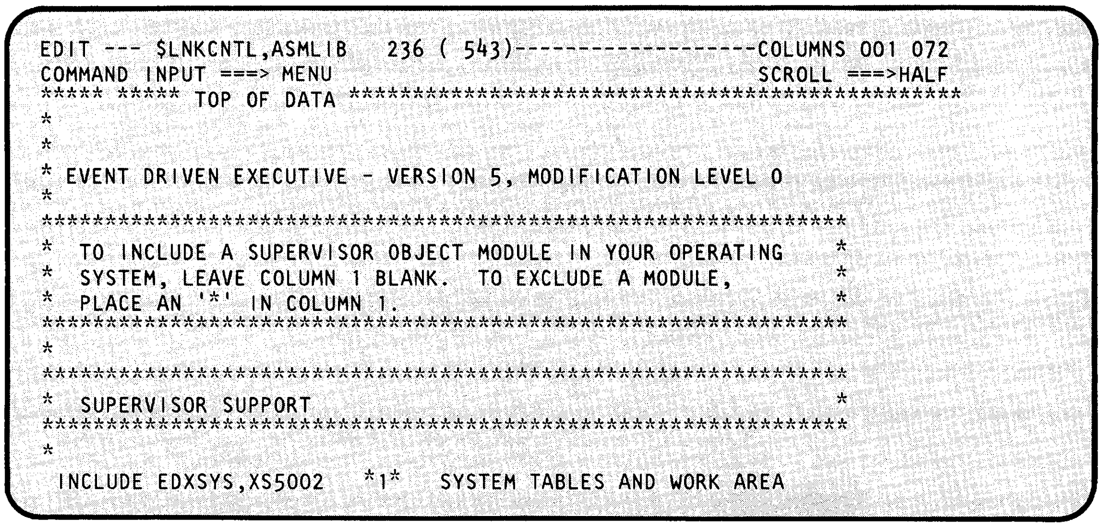
```
 OPTION NOVERLAY        *25* NO OVERLAY STRUCTURE
*---------------------------------------------------------------
* SYSTEM SUPPORT -- INITIALIZATION
*---------------------------------------------------------------
 INCLUDE EDXINIT        *24* SUPERVISOR INITIALIZATION
*INCLUDE RW4963ID       *3*  4963 FIXED HEAD REFRESH SUPPORT
 INCLUDE INITMFA        *3*  MFA INITIALIZATION
 INCLUDE INITADAP       *3*  ALPA & SMIO INITIALIZATION
 INCLUDE INIT4978       *3*  4978 TERMINAL INITIALIZATION
 INCLUDE INIT4980       *3*  4980 TERMINAL INITIALIZATION
*---------------------------------------------------------------
* INSERT USER INITIALIZATION MODULES HERE
*---------------------------------------------------------------
 PART 1
 INCLUDE SAMPLE1              SAMPLE1 INITIALIZATION
 INCLUDE SAMPLE2              SAMPLE2 INITIALIZATION
 INCLUDE SAMPLE3              SAMPLE3 INITIALIZATION
*---------------------------------------------------------------
* SUPERVISOR CODE BEING MOVED OUT OF
* PARTITION 1 'MUST' BE MOVED TO HERE
*---------------------------------------------------------------
* SENSOR INPUT/OUTPUT - MUST BE GROUPED TOGETHER
                      - MAY BE INCLUDED IN PARTITION 1 to 8
*---------------------------------------------------------------
 PART 3
 INCLUDE SBCOM         *15*  BASIC SENSOR I/O SUPPORT
 INCLUDE SBAI          *3*   ANALOG INPUT SUPPORT
 INCLUDE SBAO          *3*   ANALOG OUTPUT SUPPORT
 INCLUDE SBIDO         *3*   DIGITAL INPUT/OUTPUT SUPPORT
 INCLUDE SBPI          *3*   PROCESS INTERRUPT SUPPORT
*---------------------------------------------------------------
* BISYNC COMMUNICATION - MUST BE GROUPED TOGETHER
                       - MAY BE INCLUDED IN PARTITION 1 to 8
*---------------------------------------------------------------
 PART 4
 INCLUDE BSCAM         *7*   BISYNC COMMUNICATION SUPPORT
*---------------------------------------------------------------
```

Figure 10. Example Edited Link-Control Data Set

After you modify the link-control data set to match your work sheet, you need to save the changes. You may wish to delete the notes at the end of the data set. By deleting the notes, you will reduce the time for listing the output generated by $XPSLINK processing.

Step 4 - Edit Link Control Data Set to Include Software Support *(continued)*

4(d) **Enter MENU on the COMMAND INPUT line and press the enter key to end the EDIT mode.**



```
EDIT --- $LNKCNTL,ASMLIB   236 ( 543)------------------COLUMNS 001 072
COMMAND INPUT ===> MENU                                SCROLL ===>HALF
***** ***** TOP OF DATA ************************************************
*
*
* EVENT DRIVEN EXECUTIVE - VERSION 5, MODIFICATION LEVEL 0
*
***************************************************************
*  TO INCLUDE A SUPERVISOR OBJECT MODULE IN YOUR OPERATING    *
*  SYSTEM, LEAVE COLUMN 1 BLANK.  TO EXCLUDE A MODULE,        *
*  PLACE AN '*' IN COLUMN 1.                                  *
***************************************************************
*
***************************************************************
*  SUPERVISOR SUPPORT                                         *
***************************************************************
*
 INCLUDE EDXSYS,XS5002   *1*   SYSTEM TABLES AND WORK AREA
```

$FSEDIT returns to the primary command menu.

4(e) **Select option 4 (WRITE) to save the changes made to the link-control data set.**

You may want to maintain more than one supervisor. For example, you may need one system for development and one for production. It is recommended that you save the link-control data set that is unique for each supervisor in a separate data set. The name assigned to the data set should be named LINKCTLx, where x is any alphanumeric character.

1. Enter a **4** on the OPTION line
2. Enter name of the data set where you want to save the new link-control data set. (LINKCNTL in the example.)
3. Enter the name of the volume containing the data set. (EDX002 in the example.)
4. Press the enter key.
5. Enter a **Y** in response to the verification prompt.
6. Press the enter key.

Step 4 - Edit Link Control Data Set to Include Software Support *(continued)*

```
$FSEDIT PRIMARY OPTION MENU -----------------------STATUS = MODIFIED
                                                   PRESS PF3 TO EXIT
OPTION ===> 4

DATA SET NAME =========> LINKCNTL        (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> EDX002

    .
    .

WRITE TO LINKCNTL ON EDX002 (Y/N)? Y
```

$FSEDIT issues the following message:

```
nn LINES WRITTEN TO LINKCNTL,EDX002
```

where nn indicates the number of lines in the data set.

Example: The following is an example of the LINKCNTL data set for the sample system. Only the modules without an asterisk in column one are included in the supervisor.

```
*OPTION NOVERLAY        *25* NO OVERLAY STRUCTURE
*----------------------------------------------------------------------
*  SUPERVISOR SUPPORT    - MUST BE FIRST AND IN PARTITION 1
*----------------------------------------------------------------------
 PART 1
 VOLUME  XS5002              DEFAULT VOLUME FOR INCLUDE MODULES
*OVLAREA OVLSTART OVLEND *23*  USER DEFINED OVERLAY AREA
 INCLUDE EDXSYS          *1*   SYSTEM TABLES AND WORK AREAS
 INCLUDE ASMOBJ,EDX002   *1*   OUTPUT FROM USER SYSTEM GENERATION
 INCLUDE EDXSVCX         *1*   TASK SUPERVISOR
 INCLUDE EDXALU          *1*   EDL INSTRUCTION EMULATOR
 INCLUDE EDXSTART        *1*   INITIALIZATION & ERROR HANDLER
*INCLUDE SWAITM          *3*   WAIT ON MULTIPLE EVENTS
*----------------------------------------------------------------------
*  OPTIONAL FUNCTION SUPPORT - MUST BE IN PARTITION 1
*----------------------------------------------------------------------
 INCLUDE RLOADER         *17*  RELOCATING PROGRAM LOADER
*----------------------------------------------------------------------
*  DISK(ETTE) SUPPORT    - MUST BE GROUPED TOGETHER IN ANY PARTITION
*----------------------------------------------------------------------
```

Figure 11 (Part 1 of 2). LINKCNTL Data Set for the Sample System

Step 4 - Edit Link Control Data Set to Include Software Support *(continued)*

```
*---------------------------------------------------------------
 PART 1
 INCLUDE DISKIO          *3*   BASIC DISK(ETTE) SUPPORT
 INCLUDE D49624          *3*   4962/4964 DISK(ETTE) SUPPORT
*---------------------------------------------------------------
*  SYSTEM INITIALIZATION - MUST BE IN PARTITION 1
*---------------------------------------------------------------
 INCLUDE EDXINIT         *24*  SUPERVISOR INITIALIZATION
*---------------------------------------------------------------
*  SUPERVISOR CODE BEING MOVED OUT OF
*  PARTITION 1 "MUST" BE MOVED TO HERE
*---------------------------------------------------------------
*
*---------------------------------------------------------------
*  TERMINALS             - MUST BE GROUPED TOGETHER IN ANY PARTITION
*---------------------------------------------------------------
 PART 8
 INCLUDE EDXTIO           *4*   BASIC TERMINAL SUPPORT
 INCLUDE FULLMSG          *5*   FULL MESSAGE SUPPORT
 INCLUDE IOS4979          *3*   4978/4979 DISPLAY SUPPORT
 INCLUDE IOS4974          *3*   4973/4974/4975 PRINTER SUPPORT
*-------------------------------------------------------
LINK $EDXNUC2,EDX002 REPLACE END
*-------------------------------------------------------
*
```

Figure 11 (Part 2 of 2). LINKCNTL Data Set for the Sample System

Step 5 - Edit $JOBUTIL Procedure File

You are now ready to assemble the system definition statements and link edit the resulting object module with the supervisor support object modules specified in the edited link-control data set.

The assemble and link edit steps are performed under control of the job stream processing utility $JOBUTIL. You could load the assembler $EDXASM, provide the data set names required, and do the assembly. Then you could do the same for $XPSLINK. But, by using $JOBUTIL, the two steps can be accomplished with a single entry.

You control the $JOBUTIL operation with a series of job control statements. For system generation, a data set called $SUPPREP is supplied on volume ASMLIB.

If you did not change the recommended names for the data sets (EDITWORK, ASMWORK, ASMOBJ, and LINKWORK) in step 2 and used the names $EDXDEFS and $LINKCNTL for the definition data set and link-control data set in steps 3 and 4, you do not need to edit $SUPPREP as described in steps 5(b) and 5(c). However, we recommend saving the $SUPPREP data set to a data set called SUPPREPS on volume EDX002 for future system generations as described in steps 5(a), 5(d), and 5(e). Once you copy $SUPPREP to SUPPREPS on EDX002, proceed to step 6.

Step 5 - Edit $JOBUTIL Procedure File *(continued)*

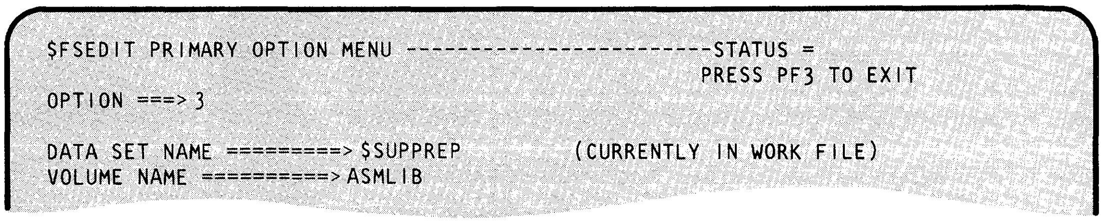
For example, if you called the assembler work data set ASMOBJ1 instead of ASMOBJ and increased its size, you would have to change the name and record size in statement 140 and the name in DS statement 220. See Figure 12 on page IS-101 which shows the $SUPPREP data set.

To edit $SUPPREP:

5(a) Read the $SUPPREP data set into the work data set.

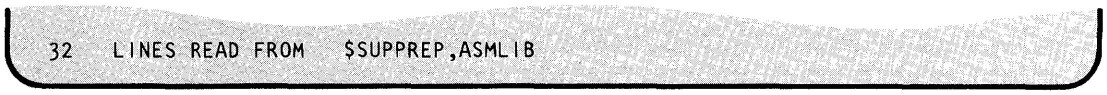
1. Enter a **3** on the OPTION line
2. Enter **$SUPPREP** on the DATA SET NAME line.
3. Enter **ASMLIB** on the VOLUME NAME line.
4. Press the enter key.

If you do not need to edit $SUPPREP, skip to step 5(d).



```
$FSEDIT PRIMARY OPTION MENU -----------------------STATUS =
                                                  PRESS PF3 TO EXIT
OPTION ===> 3

DATA SET NAME =========> $SUPPREP         (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> ASMLIB
```

$FSEDIT reads the requested data set into the EDITWORK data set and issues the following message:

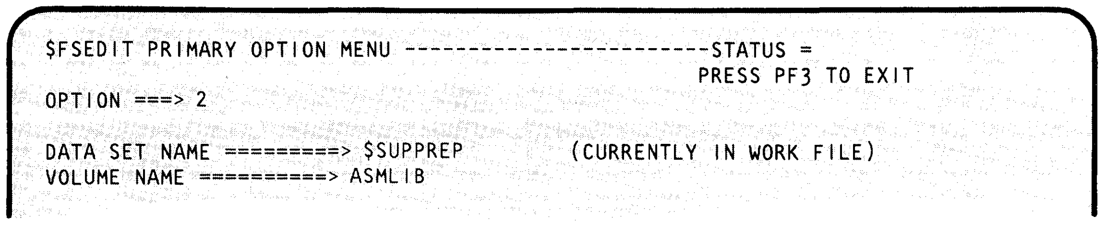


```
32   LINES READ FROM    $SUPPREP,ASMLIB
```

5(b) Select option 2 (EDIT) to edit $SUPPREP.

1. Enter a **2** on the OPTION line
2. Enter **$SUPPREP** on the DATA SET NAME line.
3. Enter **ASMLIB** on the VOLUME NAME line.
4. Press the enter key.

Step 5 - Edit $JOBUTIL Procedure File *(continued)*



```
$FSEDIT PRIMARY OPTION MENU -----------------------STATUS =
                                                  PRESS PF3 TO EXIT
OPTION ===> 2

DATA SET NAME =========> $SUPPREP        (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> ASMLIB
```

$FSEDIT displays the $SUPPREP data set.

```
EDIT --- $SUPPREP,ASMLIB    30 ( 543)------------------COLUMNS 001 072
COMMAND INPUT ===>                                     SCROLL ===>HALF
***** ***** TOP OF DATA ************************************************
00010 *
00020 *                     EVENT DRIVEN EXECUTIVE
00030 *                 VERSION 5, MODIFICATION LEVEL 0
00040 *
00050 LOG       $SYSPRTR
00060 JOB       $SUPPREP
00070 REMARK    ** ENTER -GO- AFTER -XS5002- HAS BEEN VARIED ONLINE **
00080 REMARK    ** EXECUTION PRODUCES $EDXNUCT ON EDX002 **
00090 PAUSE
00100 *         DELETE WORK DATA SETS AND REALLOCATE (OR ALLOCATE)
00110 DE        ASMWORK,EDX002
00120 AL        ASMWORK,EDX002 500 D
00130 DE        ASMOBJ,EDX002
00140 AL        ASMOBJ,EDX002 300 D
00150 DE        LINKWORK,EDX002
00160 AL        LINKWORK,EDX002 600 D
00170 PROGRAM   $EDXASM,ASMLIB
00180 NOMSG
00190 PARM      OVERLAY 4 EN
00200 DS        $EDXDEFS,EDX002
00210 DS        ASMWORK,EDX002
00220 DS        ASMOBJ,EDX002
00230 EXEC
00240 JUMP      ENDJOB,NE,-1
00250 PROGRAM   $XPSLINK,EDX002
00260 NOMSG
00270 PARM      $SYSPRTR YES
00280 DS        LINKWORK,EDX002
00290 DS        LINKCNTL,EDX002
00300 EXEC
00310 LABEL     ENDJOB
00320 EOJ
```

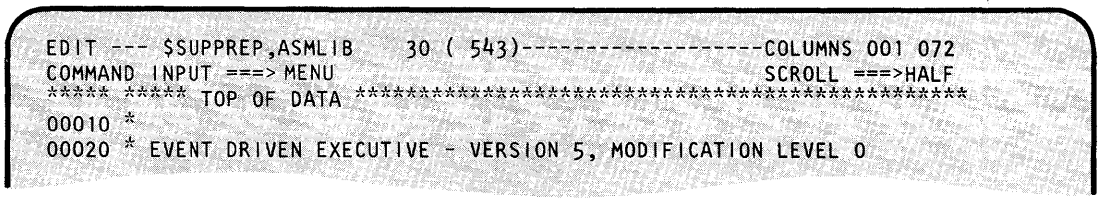
Step 5 - Edit $JOBUTIL Procedure File *(continued)*

5(c) Change the data set names and sizes where appropriate. You must change one or more of the following names:

- ASMWORK
- LINKWORK
- ASMWORK
- ASMOBJ
- $EDXDEFS
- LINKCNTL.

Some of these names appear on more than one line.

5(d) **Enter MENU in the COMMAND INPUT line to end the EDIT mode and press the enter key.**



```
EDIT --- $SUPPREP,ASMLIB    30 ( 543)------------------COLUMNS 001 072
COMMAND INPUT ===> MENU                                 SCROLL ===>HALF
***** ***** TOP OF DATA ***********************************************
00010 *
00020 * EVENT DRIVEN EXECUTIVE - VERSION 5, MODIFICATION LEVEL 0
```

$FSEDIT returns to the primary option menu.

5(e) **Select option 4 (WRITE) to save the changes made to $SUPPREP.**

1. Enter a **4** on the OPTION line.
2. Enter the new data set name, **SUPPREPS** on the DATA SET NAME line.

   Assign a unique name to this data set, if you have more than one supervisor and want to keep the $JOBUTIL data set for each. Name the data set SUPPREPx, where x is any alphanumeric character.

Step 5 - Edit $JOBUTIL Procedure File *(continued)*

3. Enter the volume name, **EDX002** on the VOLUME NAME line.

4. Press the enter key.

```
$FSEDIT PRIMARY OPTION MENU --------------------- STATUS = MODIFIED
                                                   PRESS PF3 TO EXIT
OPTION ===> 4

DATA SET NAME =========> SUPPREPS        (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> EDX002

     .
     .
WRITE TO SUPPREPS ON EDX002 (Y/N)? Y
```

$FSEDIT issues the following message,

where nn indicates the number of lines in the data set:

```
nn LINES WRITTEN TO SUPPREPS,EDX002
```

5(f) **Select option 8 (END) and press the enter key to end the $FSEDIT utility.**

```
$FSEDIT PRIMARY OPTION MENU --------------------- STATUS = MODIFIED
                                                   PRESS PF3 TO EXIT
OPTION ===> 8

DATA SET NAME =========> SUPPREPS        (CURRENTLY IN WORK FILE)
VOLUME NAME ==========> EDX002
```

$FSEDIT responds as follows:

```
$FSEDIT  ENDED
```

Step 6 - Execute $SUPPREP

To create a tailored supervisor, you must first assemble your system configuration statements in $EDXDEFS. Then you need to link edit the supervisor object modules in the link-control data set ($LNKCNTL) and convert the results to an executable program using $XPSLINK. $XPSLINK is a link editor set up for generating operating systems. There are two ways to run $XPSLINK and the procedure in SUPPREPS. You can use either the session manager option 2.8 or 2.13 or the $JOBUTIL job stream processor.

To use the session manager, you must have installed the program preparation modules (5719-XX6). For general information on how to use the session manager, see the *Operation Guide*.

The procedure described here uses the $JOBUTIL job stream processor to execute the procedure in SUPPREPS.

6(a) **Press the attention key, enter $L $JOBUTIL and press the enter key.**

```
> $L $JOBUTIL
DS1 (NAME,VOLUME): SUPPREPS,EDX002
LOADING $JOBUTIL      4P,00:51:57, LP= 9200, PART= 1
REMARK   ** ENTER -GO- AFTER -XS5002- HAS BEEN VARIED ONLINE **
REMARK   ** PRODUCES $EDXNUCT ON EDX002 **

PAUSE-*-ATTN:GO/ENTER/ABORT

PAUSE
```

6(b) **Insert diskette XS5002 into your diskette unit.**

Diskette XS5002 contains the supervisor object modules which are accessed only during the system generation process.

Note: If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. To do this:

- Press the attention key
- Press the enter key
- Enter **$VARYON nn,1**, where nn is the hexadecimal address of the 4966 Diskette Magazine Unit.

```
> $VARYON 22,1
XS5002 ONLINE ON SLOT 1
```

Step 6 - Execute $SUPPREP *(continued)*

6(c) Press the attention key and enter GO in response to the PAUSE prompt.

```
> GO
   .
   .
   .
$JOBUTIL ENDED
```

The procedure file has $SYSPRTR specified as the log device. The first thing that happens is that the procedure file statements controlling the assembly operation print out on the $SYSPRTR. $JOBUTIL then loads the compiler, $EDXASM, which assembles your system definition source file, $EDXDEFS.

The resulting object module is stored in data set ASMOBJ on volume EDX002, which you created. The listing produced as a result of the assembly prints out on the $SYSPRTR, preceded by assembler statistics.

Next, $JOBUTIL loads the linkage editor, $XPSLINK. $XPSLINK is a special linkage editor set up to generate operating systems. It loads three programs, in the following order: $XPSPRE, $EDXLINK, and $XPSPOST. Each of these programs performs its function as follows:

$XPSPRE: $XPSPRE scans the link-control data set for supervisor object modules to be located outside of partition 1. When it finds such a module, it builds separate link-control statements for each supervisor partition specified by the PART statement in the LINKCNTL data set. In addition, $XPSPRE includes a 'root' module in partition 1 that corresponds to the supervisor object module being located outside of partition 1. The 'root' segment provides the connection between the supervisor object module and the system control blocks located in partition 1.

The listing of the LINKCNTL data set produced as a result of processing $XPSPRE prints on the $SYSPRTR or the terminal you specify. A completion code of -1 indicates success.

$EDXLINK: Using the object module from the assembly (ASMOBJ) and the link-control statements stored in LINKCNTL, the $EDXLINK program link edits each group of link-control statements created by $XPSPRE with the system control blocks generated by the assembly (ASMOBJ object modules). $EDXLINK creates separate output data sets for each group of modules specified with a PART statement. These data sets are called XPSSEG1x through XPSSEG8x where x is the last character of the name of the nucleus specified on the LINK statement.

$EDXLINK prints out each data set and any unresolved references resulting from the link edit on the $SYSPRTR or the terminal you specify. There will be several unresolved weak external references (WXTRN) for supervisor support modules you did not want to include, but no unresolved EXTRN messages should appear. Following the EXTRN messages, it lists whether or not the data set containing the supervisor modules was stored, the size (in records) of the data set, and a completion code.

Step 6 - Execute $SUPPREP *(continued)*

In addition, it prints out the modules you included, the modules $XPSLINK automatically included, and the names of the root segments for each partition. At the end of this list is a total for the LNTH column. This figure is the size of the supervisor within that partition.

$XPSPOST: After $EDXLINK link edits the data sets that make up the supervisor, $XPSLINK calls $XPSPOST. $XPSPOST merges the entry point addresses for the supervisor object modules together to form a table in partition 1. This table resides in partition 1 and contains the locations of all the supervisor modules in the system. $XPSPOST also relocates the addresses of the supervisor modules outside partition 1 to allow them to be loaded at IPL time.

The formatted load module is placed in $EDXNUCx, a supervisor data set allocated automatically by $XPSLINK. $XPSLINK ends and $JOBUTIL completes.

When $JOBUTIL completes, examine the output printed on $SYSPRTR for errors. Errors are usually caused by incorrect editing of $EDXDEF, $LNKCNTL or $SUPPREP. If errors are found, examine your system definition statements and link edit control statements. Then correct $EDXDEFS, LINKCNTL, or SUPPREPS as necessary using the $FSEDIT utility.

Notes:

1. If you receive undefined label messages, you may have forgotten the continuation character in column 72.

2. Unresolved WXTRN messages resulting from the execution of $XPSLINK can occur. A WXTRN message lists modules that may or may not be required in your supervisor. Your supervisor is generated with or without these modules. Examine the message to determine whether the referenced names refer to the modules that you require in your operating system.

3. An unresolved WXTRN of $PROG1 normally occurs unless you link edit an application program with the supervisor, as described under step 3 in Chapter 4, "Select Your Required Support" on page IS-39.

4. Unresolved EXTRN messages should not occur if a valid operating system has been generated. An EXTRN message lists modules that MUST be included in your supervisor. A complete listing of all supervisor module names and entry points is included in Appendix D, "Supervisor Module Names (CSECTS)" on page IS-275. If you have not included the module listed, edit LINKCNTL to include the missing module.

Step 6 - Execute $SUPPREP *(continued)*

5. Your supervisor may have exceeded 64K within a specific partition. If it does, the following message is printed following link-control statements:

```
XPSSEG1x,volume NOT STORED
COMPLETION CODE =  12
```

The total of the LNTH column following the list of modules included in the supervisor contains the amount by which the 64K was exceeded. You must reduce the size of the supervisor by at least this amount. For information on making your supervisor smaller, see "Reducing the Size of Your Supervisor" on page IS-68.

After you correct the errors, execute $XPSLINK again through the $JOBUTIL procedure or the session manager option 2.8.

You should save the system definition statements ($EDXDEFS), the link-control statements (LINKCNTL), and the listings printed as a result of the system generation process. They provide you with a record which can be helpful if problems develop with the generated operating system.

Step 7 - Using the Generated Operating System

Before you can use the new operating system, it must be designated as the one to be loaded at IPL time. To do this, you must load $INITDSK and initialize $EDXNUCx as the new operating system. In response to the NUCLEUS prompt, enter the name of your supervisor as specified on the LINK statement in the LINKCNTL data set. If you enter a supervisor name that does not exist, $INITDSK issues the following message and ends the II command.

```
INVALID NUCLEUS NAME
```

$INITDSK issues the COMMAND (?): prompt again. Enter the II command again and reenter the name of your supervisor.

In this example, the name of the supervisor is $EDXNUC2 on EDX002.

Step 7 - Using the Generated Operating System *(continued)*

7(a) **Press the attention key and load the $INITDSK utility.** Respond to the $INITDSK prompts as shown.

```
> $L $INITDSK
 LOADING $INITDSK    98P, LP= 9200, PART= 1

 $INITDSK - DISK INITIALIZATION UTILITY

 COMMAND (?): II
 NUCLEUS: $EDXNUC2
 VOLUME: EDX002
 IPL TEXT WRITTEN

 COMMAND (?):EN
 $INITDSK ENDED AT 01:05:16
```

Now you can use $EDXNUCx as your supervisor.

7(b) **Press the LOAD button on the Series/1 console and IPL the new supervisor.**

If the new operating system fails to operate correctly, you must restore the starter system by IPLing from diskette XS5001. Then use the II command to redirect the IPL text to point to the operating system that you specify (in this case, $EDXNUC2 on EDX002). Repeat steps 2 through 5 to correct any errors.

If the IPL is successful, the IPL message is displayed. Following is the IPL message for the sample system:

```
*** EVENT DRIVEN EXECUTIVE *** V5.0
***           XPS          ***

IPL=$EDXNUC2,EDX002

                XPS SYSTEM STORAGE MAP
                ----------------------
        USER   USER  COMMON   SUPV  SUPV   USER    USER   TOTAL
PART   START   SIZE   SIZE   START  SIZE  START    SIZE   SIZE
  #    (DEC)  (DEC)  (HEX)   (HEX) (HEX)  (HEX)   (HEX)   (HEX)
  1    22272  14592     0       0   5700   5700    3900    9000
  2     4096  49152  1000    1000      0   1000    C000    D000
  3        0  32768     0       0      0      0    8000    8000
  4     2048  40960  0800    0800      0   0800    A000    A800
  5     2048  63488  0800    0800      0   0800    F800   10000
  8    11264   1024     0       0   2C00   2C00    0400    3000
TOTAL(HEX):                         8300          31500
UNMAPPED STORAGE =   137 (DEC) 2K BLOCKS
```

The IPL message contains 9 columns. An explanation of each column follows.

PART #	Part # indicates the the number of the partition.
USER START	The starting address (in decimal) of the first program loaded for execution within each partition.
USER SIZE	The amount of storage (in decimal) within each partition available for program execution.
COMMON SIZE	The size (in hexadecimal) of the common area within each partition.
SUPV START	The starting address (in hexadecimal) of the supervisor within each partition.
SUPV SIZE	The size (in hexadecimal) of the supervisor within in each partition.
USER START	The starting address (in hexadecimal) of the first program loaded for execution within each partition.
USER SIZE	The amount of storage (in hexadecimal) within each partition available for program execution.
TOTAL SIZE	The total size (in hexadecimal) of each partition.

You may now execute programs under the tailored operating system. Load and execute utility programs that exercise the various supervisor components (such as disk I/O, sensor I/O, and so forth).

If you want your supervisor named by the standard nucleus name ($EDXNUC), follow steps 6(c) and 6(d), thereby always having $EDXNUC as your current supervisor. If you want to maintain multiple supervisors, leave your supervisor named $EDXNUCx where x is a unique alphanumeric character and proceed to step 8.

Note: For information on maintaining more than one supervisor, see "Maintaining Multiple Supervisors" on page IS-113.

Step 7 - Using the Generated Operating System *(continued)*

7(c) **Press the attention key and load the $COPYUT1 utility.** Respond to the $COPYUT1 prompts as shown.

```
> $L $COPYUT1
LOADING $COPYUT1     59P, LP= 9200, PART= 1

$COPYUT1 - DATA SET COPY UTILITY


MEMBERS ON TARGET VOLUME WILL BE DELETED
REALLOCATION AND COPYING OF MEMBERS IS
DEPENDENT ON SUFFICIENT CONTIGUOUS SPACE

THE DEFINED SOURCE VOLUME IS EDX002, OK (Y/N)? Y
THE DEFINED TARGET VOLUME IS EDX002, OK (Y/N)? Y
MEMBER WILL BE COPIED FROM EDX002 TO EDX002 OK (Y/N)? Y

COMMAND (?): CM
ENTER FROM(SOURCE) MEMBER: $EDXNUC2
ENTER TO (TARGET) MEMBER OR * FOR SAME NAME AS SOURCE: $EDXNUC
COPY COMPLETE             257 RECORDS COPIED

COMMAND (?): EN
$COPYUT1 ENDED AT 00:26:07
```

8(d) **Press the attention key and load $INITDSK.** Respond to the $INITDSK prompts as shown.

```
> $L $INITDSK
LOADING $INITDSK    98P, LP= 9200, PART= 1

$INITDSK - DISK INITIALIZATION UTILITY

COMMAND (?): II
NUCLEUS: $EDXNUC
VOLUME: EDX002

COMMAND (?): EN
$INITDSK ENDED AT 01:05:16
```

The II command initializes the new supervisor (in this case, $EDXNUC2 on EDX002) so you'll always have $EDXNUC as your current supervisor.

Step 8 - Verify the System Generation Process (Optional)

To verify that system generation has performed correctly, assemble and execute the sample program CALCSRC on EDX002.

8(a) **Press the attention key, load the $EDXASM compiler, and provide the parameters necessary to assemble CALCSRC, as follows:**

```
> $L $EDXASM,ASMLIB CALCSRC ASMWORK ASMOBJ
LOADING $EDXASM      78P, LP= 9100, PART=1

SELECT OPTIONS (?):  NOLIST END
ASSEMBLY STARTED

   1 OVERLAY AREA ACTIVE
   EDX ASSEMBLER STATISTICS

   SOURCE INPUT   - CALCSRC,EDX002
   WORK DATA SET  - ASMWORK,EDX002
   OBJECT MODULE  - ASMOBJ,EDX002
   STATEMENTS PROCESSED -      66



   NO STATEMENTS FLAGGED
   EXTERNAL/UNDEFINED SYMBOLS

  SVC              WXTRN
  SUPEXIT          WXTRN
  SETBUSY          WXTRN

COMPLETION CODE =    -1

$EDXASM ENDED
```

If you assemble CALCDEMO successfully (completion code = -1), continue. If you receive a completion code other -1, see the *Messages and Codes* book for its meaning, correct the problem, and reassemble CALCDEMO.

Step 8 - Verify the System Generation Process (Optional) *(continued)*

8(b) **Press the attention key, load $EDXLINK and provide the parameters necessary to link edit CALCSRC.** This step produces a listing on the printer designated as $SYSPRTR.

Note: If you have created CALCDEMO previously, respond with LINK CALCDEMO REP END to the second STMT(?): prompt.

```
> $L $EDXLINK
 LINKWORK(NAME,VOLUME):  LINKWORK,EDX002
 LOADING $EDXLINK     91P, LP= 9100, PART=1
 $EDXLINK - EDX LINKAGE EDITOR

 PARM(?): *
 $EDXLINK INTERACTIVE MODE
 DEFAULT VOLUME - EDX002

 STMT(?): INCLUDE ASMOBJ

 STMT(?): LINK CALCDEMO END

 $EDXLINK EXECUTION STARTED
  CALCDEMO,EDX002 STORED
 PROGRAM DATA SET SIZE = 4 RECORDS
 COMPLETION CODE = -1
 $EDXLINK ENDED
```

If you receive a completion code of -1, continue. If you receive a completion code other than -1, see the $EDXLINK completion codes in *Messages and Codes* for an explanation of the code received. Correct the error and relink CALSRC.

8(c) **Press the attention key and load the verification program, CALCDEMO.** Follow its operating instructions until you decide to end the program. When you decide to end the program, enter 'STOP'.

```
> $L CALCDEMO
 CALCDEMO     3P, LP= 9100, PART=1
 PRESS 'ATTENTION' AND ENTER 'CALC' OR 'STOP'

> CALC

 A = 30
 B = 6

 A + B =            36
 A - B =            24
 A * B =           180
 A / B =             5 REMAINDER =          0
 PRESS 'ATTENTION' AND ENTER 'CALC' OR 'STOP'
> STOP
 CALCDEMO ENDED
```

Successful program execution indicates that your operating system is also executing correctly.

Step 8 - Verify the System Generation Process (Optional) *(continued)*

Use the LS of the $IOTEST utility to list the I/O devices supported by the generated supervisor. With this list, you can verify that the supervisor does indeed support all the devices you intended it to support.

Maintaining Multiple Supervisors

You may want to maintain more than one supervisor. For example, you may need one system for development and one for production. The II command (initialize IPL text) of the $INITDSK utility allows you to use any supervisor on any disk(ette) volume. The II command initializes the supervisor that you specify.

We recommend that you keep your supervisor programs on the IPL volume EDX002. When your supervisor resides on a volume other than EDX002, it searches that volume for the system utilities. However, the system utilities reside on volume EDX002. Each time you access the system utilities, you will need to indicate the volume they reside on; in this case, EDX002.

The following example points the IPL text to $EDXNUC2 on volume EDX002.

```
COMMAND (?):  II
NUCLEUS:  $EDXNUC2
VOLUME:  EDX002
IPL TEXT WRITTEN
```

The supervisor you are using ***must*** be named $EDXNUCX, where x is any alphanumeric character.

Generating an Operating System for a Diskless System

For Series/1 systems that do not have a fixed disk or the Program Preparation Facility, generating an operating system requires the following steps:

1. Compile and link edit the supervisor for the target Series/1 on a system that supports program preparation.
2. Compile the application programs for the target Series/1.
3. Use the $INITDSK utility to initialize one or more diskettes with program space for the supervisor program, utilities, additional products and application programs.
4. Transfer the supervisor to $EDXNUC on the IPL volume of the diskette with the $COPYUT1 utility. Use $INITDSK to write IPL text for $EDXNUC on the diskette.

Generating an Operating System for a Diskless System *(continued)*

5. Along with your application programs, copy the following support modules and utilities onto the IPL volume with the $COPYUT1 utility:

Support modules	$LOADER, $4978CS0, $4978IS0, $4980CS0, $4980IS0, $MFARAM, $ACCARAM, $FPCARAM, $RAMSEC, $SMIOR1
Utilities	$DISKUT1, $DISKUT2, $COPYUT1, $DASDI, $INITDSK, $TERMUT1, $TERMUT2, $TERMUT3, $IOTEST

Notes:

a. If you are going to support 1024 byte/sector diskettes copy $IO1024.

b. If you copy $DISKUT2, then copy $LOGUT00.

c. If you copy $DASDI, then copy $IDSKETT.

d. If you copy $IMAGE, then use $COPYUT1 and copy generic (CG) for $IM.

e. If you copy $FSEDIT, then use $COPYUT1 and copy generic (CG) for $FS.

f. For specific application programs, you may need modules. Refer to the appropriate program directories for these requirements.

g. For Pascal applications, the diskette must be initialized as a multivolume type if the Pascal runtime error messages are desired. The PCSMSG data set resides on the volume PASCAL.

h. If you copy $TERMUT1, then use COPYUT1 and copy generic (CG) for $TERM.

6. Install the diskette(s) on the target machine for execution.

For each unique Series/1 supervisor that you create, save the associated system definition statements ($EDXDEFX), the link-control statements (LINKCNTL) with different names, and the listing printed as a result of the system generation process. They provide you with a record which can be helpful if problems develop with the generated operating system.

Multifunction Attachment Random Access Memory

The random access memory module for the Multifunction Attachment (MFA) Feature #1310 is provided with the Event Driven System (EDX) and with a Program Temporary Fix (PTF), if necessary. The module $MFARAM on volume EDX002 is a 45 record data set containing three 15-record random access memory load modules, one for each engineering level of the MFA card. EDX loads the appropriate module when you IPL your system.

If the microprocessor within the MFA is changed because of an engineering change, an updated random access memory module ($$EDXLIB,MFA) is included on the Customer Service Representative's initializer diskette. Use the $COPY utility to copy the 15-record module ($$EDXLIB,MFA) to one of the three 15-record segments of $MFARAM on EDX002 as determined by the engineering level of the MFA card.

To copy the MFA initializer diskette:

1. Insert the initializer diskette (volume MFA) into the diskette unit and close the door.

2. Use the $VARYON command to vary the initializer diskette (volume MFA) online.

3. IPL your system.

4. Load the $COPY utility and use the CD command to copy the updated module from the initializer diskette to the appropriate 15 records of $MFARAM on EDX002.

The following examples show using $COPY (CD command) to copy the initializer diskette. Respond as shown when copying the diskette.

Example 1: Copy the first 15 records to $MFARAM,EDX002.

```
> $L $COPY
LOADING $COPY     46P, LP=7E00, PART=1

$COPY - COPY UTILITY

COMMAND (?):  CD $$EDXLIB,MFA
FIRST RECORD:  1
LAST RECORD:  15
TARGET(NAME,VOLUME):  $MFARAM,EDX002
FIRST RECORD:  1
ARE ALL PARAMETERS CORRECT (Y/N)?  Y
COPY COMPLETE
          15 RECORDS COPIED

COMMAND (?):
```

Multifunction Attachment Random Access Memory *(continued)*

Example 2: Copy the second 15 records to $MFARAM,EDX002.

```
> $L $COPY
LOADING $COPY     46P, LP=7E00, PART=1

$COPY - COPY UTILITY

COMMAND (?):  CD $$EDXLIB,MFA
FIRST RECORD:  1
LAST RECORD:  15
TARGET(NAME,VOLUME):  $MFARAM,EDX002
FIRST RECORD:  16
ARE ALL PARAMETERS CORRECT (Y/N)?  Y
COPY COMPLETE
          15 RECORDS COPIED

COMMAND (?):
```

Example 3: Copy the third 15 records to $MFARAM,EDX002.

```
> $L $COPY
LOADING $COPY     46P, LP=7E00, PART=1

$COPY - COPY UTILITY

COMMAND (?):  CD $$EDXLIB,MFA
FIRST RECORD:  1
LAST RECORD:  15
TARGET(NAME,VOLUME):  $MFARAM,EDX002
FIRST RECORD:  31
ARE ALL PARAMETERS CORRECT (Y/N)?  Y
COPY COMPLETE
          15 RECORDS COPIED

COMMAND (?):
```

Chapter 6. Migrate to Version 5

This chapter describes what you must do to convert from Version 1 or 2 to Version 5. Both programs and data must be converted before Version 5 can become operational.

Conversion of Programs

Before you can use EDX Version 5, you must recompile or reassemble all EDX Version 1 or 2 programs. The source code must be modified if the Version 5 function is to be utilized or if the program references system control blocks.

In Version 5, the parameter area contained in the program header (defined by the PARM= operand of the PROGRAM statement) is smaller than the Version 2 parameter area. In Version 5 it is 980 bytes, less 66 bytes for each data set name or overlay name specified on the PROGRAM statement.

If an EDL program was used with 4978 or 4979 Display Terminals and will now be used with 3101 Display Terminals, then you should read the Version 5 *Language Reference* if you wish to make coding changes to the program.

Conversion of Programs *(continued)*

Conversion Requirements

The program conversion requirements are as follows:

- FORTRAN, COBOL, and PL/I applications:

  Compile the applications using the current compiler level for EDX Version 5; link edit with the application. After the data conversion is completed, the application is ready for execution. Data sets in the new format can be accessed in either sequential or direct mode.

- EDL and assembler language applications:

  Source code changes are required if the application:

  - Uses direct access or uses NOTE or POINT in sequential access and will access data records whose relative record number is greater than 32,767.
  - References fields in the DSCB, other than $DSCBNAM, $DSCBVOL, and the return code word (the label on the DSCB). Most of the fields in the DSCB have new labels and certain fields have become two words long.
  - Uses $DISKUT3. Word 3 of the request block is a doubleword in Version 5. If the rename function is being requested, the address of the new name is right adjusted in this 2-word field.
  - Accesses or modifies supervisor control blocks.

  If the application does not do any of the above, only recompilation or reassembly is required.

The new diskette sector size and the disk resident table of contents do not affect existing applications.

Conversion of Data

The directory format on disk and diskette is different in Version 5 than it was in Versions 1 and 2. This difference in directory format requires that data sets on disk and diskette written by Version 1 or 2 be converted to Version 5 format before they may be accessed by data management facilities. Three conversion aids ($MIGAID, $MIGRATE, and $MIGCOPY) and a diskette formatting program ($SSINIT) perform the conversion.

$MIGAID, executing on a Version 1 or 2 system, captures the contents of a disk volume on a series of diskettes called a ***save set***. A save set produced by $MIGAID may be used as input to either $MIGRATE (to convert the saved data sets to Version 5 format) or to $MIGAID (to restore the data sets to their original Version 1 or 2 format).

Note: To ensure data integrity when migrating from EDX Version 1 or 2 to EDX Version 5, an updated copy of $MIGAID must be added to the EDX system. The updated copy of $MIGAID can be obtained through your local IBM Systems Engineer. In Version 1, it runs on V1.3 PTF Level A (Version 1 Modification level 3) or later. In Version 2, it runs on V2.1 PTF Level 7. If you are currently running on Version 3 or are not converting to Version 5 from a previous version, you will not need this program.

A diskette initialization utility, $SSINIT, also included on the PTF diskette, initializes and verifies the diskettes that $MIGAID uses.

$MIGRATE, executing on a Version 5 system, reads the diskettes of the ***save set*** and produces, on disk, the Version 5 equivalents of the original data-type data sets captured by $MIGAID when it created the save set. Its input is a $MIGAID save set.

$MIGRATE will not convert program-type data sets since, except in special cases, Version 1 or 2 programs will not execute on Version 5 unless they are recompiled.

$MIGCOPY, executing on a Version 5 system, converts data sets on Version 1 or 2 diskettes to Version 5 data sets on either disk or diskette. $MIGCOPY uses many of the commands that $COPYUT1 uses and functions in much the same way. $MIGCOPY will not process program-type data sets. $MIGCOPY has an 'LA' command to list the contents of Version 1 or 2 format diskettes.

To convert to the EDX Version 5 data management environment:

1. Create an IPLable diskette containing the Version 1 or 2 $EDXNUC and starter utilities.

2. Create a save set from the Version 1 or 2 disk (4962 or 4963) environment with $MIGAID.

3. After installing EDX Version 5, recover the relevant data sets from the save set with $MIGRATE.

4. Convert Version 1 or 2 format diskettes as the need arises using $MIGCOPY.

You may use $TAPEUT1 to accomplish the migration if you have a 4969 tape unit. This procedure is described in "Disk Conversion Procedure Using Tape" on page IS-124.

The $MIGAID save set and the IPLable diskettes are also the means for returning to the original environment, should that need arise.

Conversion of Data *(continued)*

Disk Conversion Procedure Using Diskettes Produced by $MIGAID

Use this procedure only if:

1. You have installed a replacement $MIGAID module from a diskette dated October, 1983 or later. This diskette can be obtained from your IBM Systems Engineer.

2. The system you are migrating from is at EDX Version 1 Modification Level 3, PTF P0A or EDX Version 2 Modification Level 1, PTF P07.

If these conditions cannot be satisfied, an alternate procedure using $COPYUT1, $COPY, and $MIGCOPY is described later in this chapter.

The procedure to follow, to ensure that you can return to Version 1 or 2 as well as to proceed to Version 5, is:

1. Make an IPLable Version 1 or 2 diskette with the minimum starter utilities. If you have two diskette drives, you may use $COPYUT1 to copy the distributed starter system diskette (UT3001 or UT4001). If not, you will need to construct a diskette containing the minimum starter utilities.

   In either case, proceed as follows:

   a. Use $VARYON to vary the diskette online.

   b. Use $INITDSK to initialize the diskette you will be using.

      1) Use the ID command to write the volume label and the owner id with names of your choice.
      2) Allocate a directory of 60 records.
      3) Allocate a 64K $EDXNUC.
      4) Write IPL text.

   c. Use $DISKUT1 (LA command) to list all supervisor programs. You may maintain multiple supervisor programs and need to decide which versions you want to save.

   d. Use $COPYUT1 to copy your $EDXNUCx on disk to diskette.

   If you have two diskette drives, use the $COPYUT1 CALL command to copy all the data sets on the starter system diskette to the diskette you are constructing.

Conversion of Data *(continued)*

If you have only one diskette unit, use the $COPYUT1 CM command to copy individually the following from the IPL volume on disk to the diskette:

```
$4978ISO     $IOTEST
$4978CSO     $INITDSK
$4978CS1     $LOADER
$COPYUT1     $LOGUT00
$DASDI       $I4962
$DISKUT1     $I4963
$DISKUT2     $TERMUT1
$IDSKETT
```

2. Copy $MIGAID to this diskette.

3. To ensure that you can reconstruct your system, IPL from this diskette.

4. **Save this diskette**! This is your only vehicle for rebuilding your current disks.

5. IPL the disk and allocate the $MIGAID work file (see example, below). Do *not* allocate your work file in the volume you are going to save.

6. Use $SSINIT to initialize the diskettes for use by $MIGAID. The number of diskettes required to save the contents of a volume depends upon the amount of data allocated on that volume.

7. Use $MIGAID to save the contents of your volumes on diskettes.

8. Install the Event Driven Executive using the procedures outlined in the Version 5 Program Directories.

9. Run $MIGRATE to copy the diskettes to your Version 5 disks.

Examples of this procedure are given below.

To return to the Version 1 or 2 environment, proceed as follows:

1. IPL the diskette you built in steps 1, 2, and 3, above.
2. Execute $INITDSK to initialize your volumes on disk.
3. Allocate the $MIGAID work file on disk.
4. Load $MIGAID and execute the restore (RE) function.

You may remove the IPL diskette at this time; insert the saved diskettes as prompted by $MIGAID.

Conversion of Data *(continued)*

Disk Conversion Procedure Using Diskettes Produced by $COPYUT1 or $COPY

The procedure to follow, to ensure that you can return to Version 1 or 2 as well as to proceed to Version 5, is:

1. Make an IPLable Version 1 or 2 diskette with the minimum starter utilities. If you have two diskette drives, you may use $COPYUT1 to copy the distributed starter system diskette (UT3001 or UT4001). If not, you will need to construct a diskette containing the minimum starter utilities.

   In either case, proceed as follows:

   a. Use $VARYON to vary the diskette online.

   b. Use $INITDSK to initialize the diskette you will be using.

      1) Use the ID command to write the volume label and the owner id with names of your choice.
      2) Allocate a directory of 60 records.
      3) Allocate a 64K $EDXNUC.
      4) Write IPL text.

   c. Use $DISKUT1 (LA command) to list all supervisor programs. You may maintain multiple supervisor programs and need to decide which versions you want to save.

   d. Use $COPYUT1 to copy your $EDXNUCx on disk to diskette.

   If you have two diskette drives, use the $COPYUT1 CALL command to copy all the data sets on the starter system diskette to the diskette you are constructing.

   If you have only one diskette unit, use the $COPYUT1 CM command to individually copy the following from the IPL volume on disk to the diskette:

```
$4978IS0     $IOTEST
$4978CS0     $INITDSK
$4978CS1     $LOADER
$COPYUT1     $LOGUT00
$DASDI       $I4962
$DISKUT1     $I4963
$DISKUT2     $TERMUT1
$IDSKETT     $COPY
```

2. To ensure that you can reconstruct your system, IPL from this diskette.

3. **Save this diskette!** This is your only vehicle for rebuilding your current disks.

4. IPL the disk.

5. Use $DISKUT1 (LA command) to list the names and sizes (in 256 byte records) of all the data sets on each volume on your disk. You may want to refer to this list so obtain a hard copy of it.

6. Use the $INITDSK and DASDI utilities to prepare a supply of diskettes to hold the data included for conversion. Use a systematic sequence of volume names to assist you in keeping the diskettes in order.

7. Use $COPYUT1 to copy each data set from disk to diskette (in Version 1 or 2 format). If any data sets are too large to fit on a diskette, keep a record of its name, and continue the copy process.

8. To copy a data set that is too large to be copied in the previous step, break the data set into portions and copy each portion to a diskette using the $COPY utility. To do this:

   a. Use $INITDSK to initialize each diskette with a directory for one data set.

   b. Use $DISKUT1 to allocate the largest possible data set on each diskette prior to copying the data set. If you are using a Diskette 1, the largest data set you can allocate is 946 records; for a Diskette 2, the largest data set is 1921 records.

   c. Keep track of the number of records you copy to each diskette. In addition, keep track of the order and number of diskettes you used to copy to each data set. You will need this information when the data set is copied and reassembled into one data set on the target system.

9. Install the Event Driven Executive using the procedures outlined in the EDX Version 5 Program Directory.

10. Run $MIGCOPY to copy your Version 1 or 2 diskettes to your Version 5 disk.

11. Consider copying the fragmented data sets that were created by $COPY first. To do this:

    a. Use $MIGCOPY to copy all diskette portions of a data set to disk.

    b. Allocate a disk data set of the original name and size.

Conversion of Data *(continued)*

   c. Use $COPY to reassemble the data set by record numbers.

   d. Delete the data set portions from disk, leaving the assembled data set.

   e. Repeat steps 1 through 4 for each fragmented data set.

12. To return to the Version 1 or 2 environment:

   a. IPL the diskette built in steps 1 and 2 above.

   b. Initialize your volumes on disk using $INITDSK.

   c. Copy your Version 1 or 2 diskettes back to your Version 1 or 2 disk using $COPYUT1. (This procedure is identical to step 10, above, except that $COPYUT1 is used to copy data set instead of $MIGCOPY).

Disk Conversion Procedure Using Tape

To use tape as your conversion medium, proceed as follows:

1. Save the Version 2 disks for backup:

   a. Create a diskette that you can IPL by using the initialize device command (ID) of the $INITDSK utility. Then use the copy member command (CM) of $COPYUT1 to copy $LOADER and the copy generic command (CG) to copy the $TAPExx members. $INITDSK and $COPYUT1 are described in the *Operator Commands and Utilities Reference.*

   b. Use the save tape (ST) function of $TAPEUT1 to save the contents of each disk device.

   If you need to restore your Version 2 system, IPL the diskette created in 1.a above, then use the tapes created in 1.b above to re-create the Version 2 disks. The $TAPEUT1 restore tape (RT) command is used to restore from tape to disk.

2. Save selected data sets for use with Version 5:

   a. Use the copy data set (CD) function of $TAPEUT1 to copy the data sets to tapes that you wish to use (or have available for use) in your Version 5 system.

   b. After you have installed Version 5, use the Version 5 $TAPEUT1 copy data set (CD) command to copy the data sets from tape to disk, using the tapes created in 2.a above.

Of course, you must use $MIGCOPY to convert diskettes.

Diskette Conversion Procedure

To convert diskettes:

1. After you install Version 5, use $INITDSK to allocate and initialize a 2000-record disk work volume.
2. Use $MIGCOPY to copy the diskette to be converted to the work volume.
3. Save the diskette (see Note on the following page).
4. Initialize a different diskette, using $INITDSK.
5. Use the copy all (CALL) function of $COPYUT1 to copy the contents of the work volume to the diskette created in number 4 above.
6. Use $INITDSK to delete the work volume.

If your system has two diskette devices, you may copy from diskette to diskette by following this procedure:

1. Use $INITDSK to initialize a diskette.
2. Use $MIGCOPY to copy from your diskette to the new diskette.

Note: You should not destroy the Version 1 or 2 diskettes or the diskettes built by $MIGAID until you are satisfied that you have:

- Finished migrating all data.
- Completed conversion from Version 1 or 2 to Version 5.

Once these diskettes are reused, you have lost your ability to recover data or to return to the Version 1 or 2 operating environment.

Conversion Utilities

Three conversion aids ($MIGAID, $MIGRATE, and $MIGCOPY) and a diskette formatting program ($SSINIT) perform the conversion. A description of each follows.

$MIGAID and $SSINIT

$MIGAID captures the contents of each Version 1 or 2 volume you elect to migrate by writing them to diskettes. $SSINIT initializes these diskettes. With these diskettes, you can either restore your current environment or place selected contents of the diskettes on your Version 5 disks.

Note: Use $MIGAID and $SSINIT on your EDX Version 1 or 2 system only if:

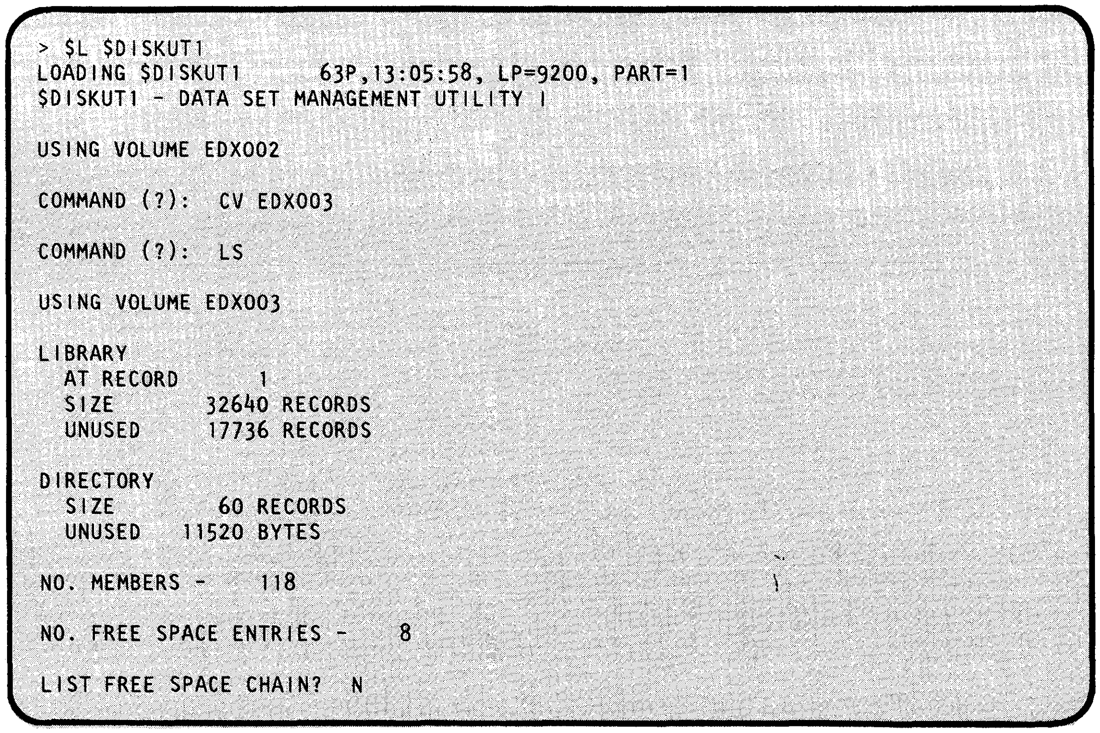
1. You have installed a replacement $MIGAID module from a diskette dated October, 1983 or later. This diskette can be obtained from your IBM System Engineer.

2. The system you are migrating from is at EDX Version 1 Modification Level 3, PTF P0A or EDX Modification Level 1, PTF P07.

The $MIGAID save function (SA) requires a work data set that must be at least one record larger than the directory of the volume to be processed. The restore (RE) function also requires a work data set of the same size. To determine the size of the work data set, proceed as follows:

1. Load $DISKUT1.
2. Change volume (CV) to the volume to which you are migrating.
3. Issue an LS command (the directory size is displayed on your terminal).
4. Reply N to the list prompt.
5. End the utility.

For example, to obtain the directory size of EDX003:



```
> $L $DISKUT1
LOADING $DISKUT1      63P,13:05:58, LP=9200, PART=1
$DISKUT1 - DATA SET MANAGEMENT UTILITY I

USING VOLUME EDX002

COMMAND (?):  CV EDX003

COMMAND (?):  LS

USING VOLUME EDX003

LIBRARY
  AT RECORD       1
  SIZE       32640 RECORDS
  UNUSED     17736 RECORDS

DIRECTORY
  SIZE        60 RECORDS
  UNUSED   11520 BYTES

NO. MEMBERS -    118

NO. FREE SPACE ENTRIES -    8

LIST FREE SPACE CHAIN?  N
```

You have found that your work data set must be 61 records (directory size 60, plus one).

Proceed now to allocate and clear the data set, using $DISKUT1,AL and $DISKUT2,CD. *Do not* allocate this work data set on the volume you are going to process. For example, allocate MIGWORK on EDX002:

```
USING VOLUME EDX002

COMMAND (?):  AL MIGWORK 61 Y
MIGWORK CREATED

COMMAND (?):  EN

$DISKUT1 ENDED AT 13:15:57
```

To clear the data set:

```
>$L $DISKUT2
LOADING $DISKUT2      51P,13:19:11, LP= 9200, PART=1
$DISKUT2  - DATA SET MANAGEMENT UTILITY II

USING VOLUME EDX002

COMMAND (?):  CD MIGWORK
CLEAR ENTIRE DATA SET?  Y

ARE ALL PARAMETERS CORRECT?  Y
CLEAR COMPLETED

COMMAND (?):  EN

$DISKUT2 ENDED AT 13:19:38
```

Now you are ready to load $MIGAID; it will prompt for your work file name and volume. For example:

```
>$L $MIGAID
WORKFILE(NAME,VOLUME):  MIGWORK,EDX002
$MIGAID     44P,14:05:07,  LP= 0000
COMMAND (?):
```

Do not delete or clear this work file until you have completely processed an entire volume. $MIGAID records in the work file the data sets that have been processed and other internal information, allowing you to stop the process and resume without having to restart from the beginning.

Now you are ready to start the save (SA) process. For obvious reasons (data integrity, completeness, and accuracy) $MIGAID must be the only active program in your system.

Conversion Utilities *(continued)*

Displaying $MIGAID Commands

To display the $MIGAID commands at your terminal, enter a question mark (?) in response to the prompting message COMMAND (?):.

```
COMMAND (?):  ?
$MIGAID -- MIGRATION AID UTILITY

$MIGAID ACCEPTS THE FOLLOWING COMMANDS:

?  -- HELP. PRINTS THE FOLLOWING LIST OF COMMANDS
      AND EXPLANATIONS.
EN -- END. COMMAND TO END $MIGAID.
PR -- PRINTER. NAME THE LOGICAL PRINTER TO WHICH
      LOGS SHOULD GO. "PR *" MEANS THE CURRENT
      TERMINAL. DEFAULT: "PR $SYSPRTR"
RE -- RESTORE. RESTORE DATA SAVED WITH THE
      "SA" COMMAND TO A VERSION 1 or 2 SYSTEM.
SA -- SAVE. SAVE DATA FROM A VERSION 1 or 2 DISK
      VOLUME ON DISKETTE(S) FOR LATER TRANSFER
      OR RESTORE

IN ADDITION $MIGAID HAS THE FOLLOWING
'ATTENTION' COMMANDS:

STOP -- STOP A SAVE OR RESTORE THAT IS IN PROCESS
GO   -- CAUSE $MIGAID TO START AGAIN WHEN IT HAS
        PAUSED FOR MORE DISKETTES.
```

You can use these commands to perform the following functions:

EN End $MIGAID. Use EN to terminate $MIGAID. EN can be entered any time the COMMAND (?): prompt appears.

PR Assign printer. Use PR to select the device used for printing the log. The default is $SYSPRTR; entering PR * causes the log to print on your terminal. It is recommended that a hard-copy device be used because the log records the name of each data set copied, the number of records copied, and shows the contents of each diskette.

RE Use RE to restore from diskette to disk. Restore (RE) copies data sets previously saved with the $MIGAID SA command from diskette and places them on a Version 1 or 2 disk.

SA Use SA to save from disk to diskette. Save (SA) copies data sets from disk to diskette. SA also calculates and tells you the number of diskettes required to contain the specified volume. Since it saves data sets, it prints the log on the device you specified (or defaulted) in the PR command.

$MIGAID also has two attention commands. You can enter these commands after you have pressed the ATTN key, or its equivalent, and have received the > prompt. The attention commands are:

STOP Directs $MIGAID to stop processing when it reaches the end of the data set it is copying, whether the operation is an SA or an RE. When $MIGAID stops, the log is printed to show where you terminated the operation.

GO Directs $MIGAID to resume processing. Use GO when you have mounted the required diskettes.

Example 1 - Preparing to Create a Save Set: This example shows how to determine the number of diskettes required to contain LIB002, and how to initialize these diskettes.

```
COMMAND (?): SA
NAME OF VOLUME TO MIGRATE: LIB002
SAVE SET NAME (1-4 CHARS): X

MIGASDV - SAVE THE DATA FROM VOLUME LIB002
          ON SAVE SET X
MIGANDR - THIS SAVE REQUIRES 5 TWO SIDED DISKETTES.
          THE DISKETTES SHOULD BE FORMATTED FOR
          EDX BY $SSINIT AND HAVE A DIRECTORY
          120 RECORDS IN SIZE.  THEY SHOULD NOT
          HAVE SPACE FOR A NUCLEUS RESERVED.
          THE VOLUME LABELS SHOULD BE X01
          THROUGH X05.  THE SAVE WILL START WITH
          DISKETTE X01
MIGALDI - LOG WILL GO TO $SYSPRTR

 ***** DATA ON THE DISKETTES WILL BE OVERWRITTEN *****
MIGAACQ - CONTINUE (Y/N): N
COMMAND (?): EN
```

Five diskettes are required to contain LIB002. Load $SSINIT to initialize the five diskettes.

$SSINIT prompts you for:

- The number of diskettes in the save set
- The volume label to be used
- The owner identification characters
- The device address.

When you have completed these replies, $SSINIT prompts you to insert a diskette. After inserting the diskette, press the enter key. Four information messages will appear, followed by the prompt for another diskette. This process is repeated until the save set initialization and verification is completed.

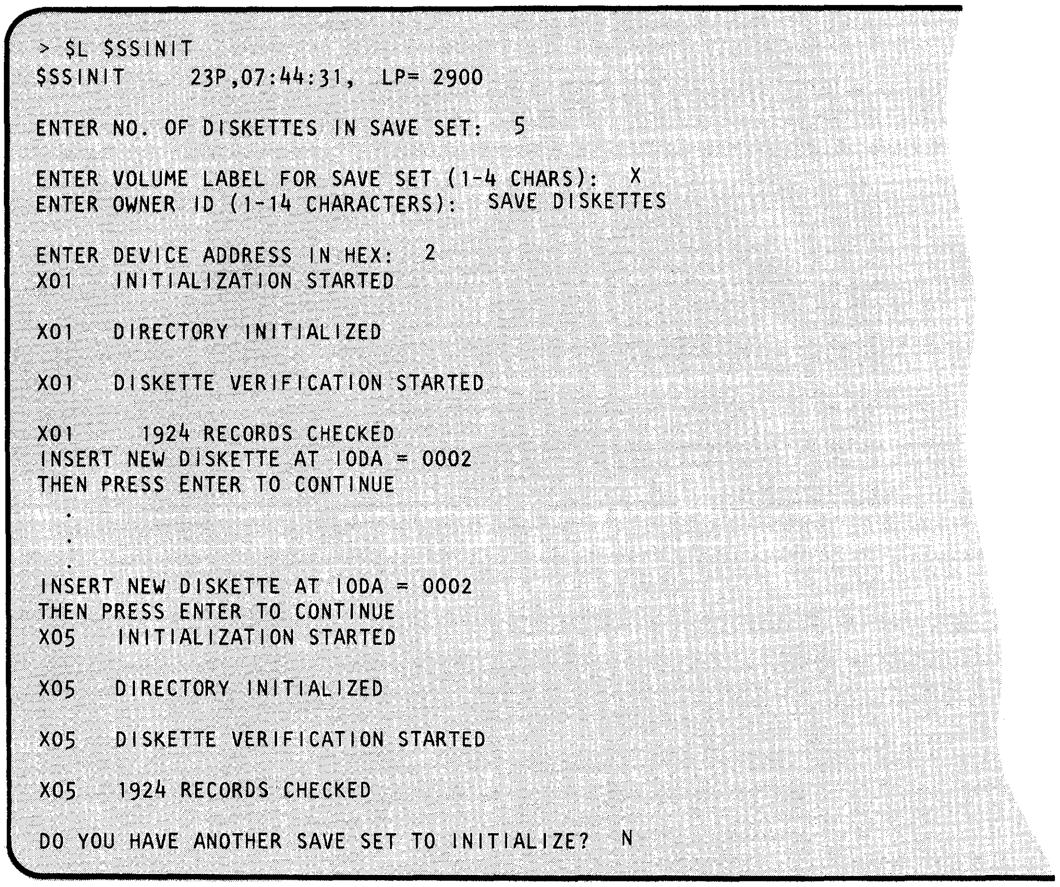
Conversion Utilities *(continued)*

As you process a diskette, $SSINIT appends a sequence number to the volume prefix you entered to create a volume label for the diskette. ***Remember to write this volume label on the external label of the diskette.*** These volume labels are your identification for these diskettes.

Do not use the EDX $VARYON/$VARYOFF commands when changing diskettes.

If you are using a 4966, each diskette must be in address slot one, as specified in the prompt message.

In this example, X01 through X05 are the volume labels of the diskette. The example shows how to initialize these diskettes:



```
> $L $SSINIT
$SSINIT     23P,07:44:31,  LP= 2900

ENTER NO. OF DISKETTES IN SAVE SET:  5

ENTER VOLUME LABEL FOR SAVE SET (1-4 CHARS):  X
ENTER OWNER ID (1-14 CHARACTERS):  SAVE DISKETTES

ENTER DEVICE ADDRESS IN HEX:  2
X01   INITIALIZATION STARTED

X01   DIRECTORY INITIALIZED

X01   DISKETTE VERIFICATION STARTED

X01     1924 RECORDS CHECKED
INSERT NEW DISKETTE AT IODA = 0002
THEN PRESS ENTER TO CONTINUE
  .
  .
  .
INSERT NEW DISKETTE AT IODA = 0002
THEN PRESS ENTER TO CONTINUE
X05   INITIALIZATION STARTED

X05   DIRECTORY INITIALIZED

X05   DISKETTE VERIFICATION STARTED

X05   1924 RECORDS CHECKED

DO YOU HAVE ANOTHER SAVE SET TO INITIALIZE?  N
```

The process of initializing and verifying these five diskettes should take approximately five minutes.

When the save set initialization is complete, you are ready to start the save process.

If an I/O error occurs while processing a diskette, the following messages appear on your terminal:

```
ERROR       5 AT RECORD    222
X01         195 RECORDS CHECKED

CONTINUE WITH INITIALIZATION OF THIS SAVE SET?
```

In this example, the READ/WRITE return code is 5, and the failure occurred at record 222 on the X01 diskette.

$SSINIT allows you to either continue or end the save set initialization. To end, reply with an N; to continue reply with a Y. The following example shows the continue option.

```
INSERT NEW DISKETTE AT IODA   =  0002
THEN PRESS ENTER TO CONTINUE

X01  INITIALIZATION STARTED

X01  DIRECTORY INITIALIZED

X01  DISKETTE VERIFICATION STARTED

X01  1924 RECORDS CHECKED
```

$SSINIT resumes with the failing volume label.

Example 2 - Create a Save Set: This example shows the sequence of events necessary to actually carry out the save prepared for in example 1. It then examines the log produced during the operation. The example begins with the save (SA) command of $MIGAID.

```
COMMAND (?):  SA
VOLUME TO MIGRATE ("*" FOR CURRENT DEFINITION):  *
(This prompt not the same as the one in Example 1)

MIGASDV - SAVE THE DATA FROM VOLUME LIB002 ON SAVE
          SET X
MIGANDR - THIS SAVE REQUIRES 5 TWO SIDED DISKETTES.
          THE DISKETTES SHOULD BE FORMATTED FOR EDX
          BY $SSINIT AND HAVE A DIRECTORY 120 RECORDS
          IN SIZE. THEY SHOULD NOT HAVE SPACE FOR A
          NUCLEUS RESERVED. THE VOLUME LABELS SHOULD
          BE X01 THROUGH X05.  THE SAVE WILL START
          WITH DISKETTE X01
MIGALDI - LOG WILL GO TO $SYSPRTR

 ***** DATA ON THE DISKETTES WILL BE OVERWRITTEN *****
MIGAACQ - CONTINUE? (Y/N):  Y
MIGAOSC - OPERATION SUCCESSFULLY COMPLETED.
COMMAND (?):
```

In this example, the save set name used is X. Since $MIGAID has determined that five diskettes are required, diskettes labeled X01 through X05 will be required. This save requires approximately 12 minutes.

The log below shows the name and size of each data set. In this example, the diskettes were mounted in the A magazine of a 4966, so there is no manual intervention required to change diskettes.

The contents of the log ($SYSPRTR) are:

```
MIGASHM - LOG OF SAVE FROM VOLUME LIB002
MIGACDS - DATA SETS COPIED FROM LIB002 TO X01
  DLETEMP1(PART)
MIGACDS - DATA SETS COPIED FROM LIB002 TO X02
  DLETEMP1  1805    DLETEMP    1803
MIGACDS - DATA SETS COPIED FROM LIB002 TO X03
  DLETEMP2(PART)
MIGACDS - DATA SETS COPIED FROM LIB002 TO X04
  DLETEMP2  1806    DLETEMP3    1802
MIGACDS - DATA SETS COPIED FROM LIB002 TO X05
  DMINTRFS 273  DMINTRFO 60 $MIGAID 64 $MIGAIDO 100
  $MIGAIDL 150  $MIGAIDC  2 TSTUT3  42 $MIGAIDS 679

MIGADSC - OPERATION SUCCESSFULLY COMPLETED.
```

The above log means:

- The first part of DLETEMP1 is on diskette X01.
- The remainder of DLETEMP1 and DLETEMP are on diskette X02.
- The first part of DLETEMP2 is on diskette X03.
- The remainder of DLETEMP2 and DLETEMP3 are on diskette X04.
- Diskette X05 contains eight data sets.

The $MIGAID restore (RE) command will rebuild the split data sets DLETEMP1 and DLETEMP2 in their proper form, as will $MIGRATE.

Example 3 - Restoring a Volume: The following example of the restore (RE) function shows a restore of EDX003 from save set MJG. The diskettes were read from a 4964 diskette unit. This example takes approximately five minutes, including the time to replace ($VARYON) the diskette.

```
 > $L $MIGAID
WORKFILE(NAME,VOLUME):  MIGWORK
$MIGAID     53P,00:44:39, LP= 0000

COMMAND (?):  RE
VOLUME TO MIGRATE ("*" FOR CURRENT DEFINITION):  EDX003
SAVE SET NAME (1-4 CHARS):  MJG
MIGARDV - RESTORE THE DATA TO VOLUME EDX003 FROM
          SAVESET MJG THE RESTORE WILL START WITH
          DISKETTE 1
MIGALDI - LOG WILL GO TO $SYSPRTR
MIGAACQ - CONTINUE? (Y/N):  Y
MIGAPMV - MOUNT VOLUME MJG01 AND VARY IT ONLINE
          TYPE 'ATTN GO' WHEN READY
          OR 'ATTN STOP' TO ABORT MOUNT
 > $VARYON 2
MJG01 ONLINE
 > GO
MIGAPMV - MOUNT VOLUME MJG02 AND VARY IT ONLINE
          TYPE 'ATTN GO' WHEN READY
          OR 'ATTN STOP' TO ABORT MOUNT
 > $VARYON 2
MJG02 ONLINE
 > GO
MIGAOSC - OPERATION SUCCESSFULLY COMPLETED
COMMAND (?):
```

Conversion Utilities *(continued)*

The contents of the log ($SYSPRTR) are:

```
MIGARHM - LOG OF RESTORE TO VOLUME EDX003

MIGACDS - DATA SETS COPIED FROM MJG01 TO EDX003
$SM2GRIZ 400 $SM3GRIZ 250 $SMPJEFF  30 $SM1GRIZ 400
$SMPGRIZ  30 $SMWGRIZ  30 $SMEGRIZ 400 MJGP01     5
MJGSRC1   50 MJGEDIT   50 MJGWRK   100 MJGAOBJ  500
MJGP02     4 SCREEN03   3

MIGACDS - DATA SETS COPIED FROM MJG02 TO EDX003
SCREEN02   3 SCREEN01   3 DEMO1     39 DEJOBS   100
DEMAPS   100
```

Save set MJG was two diskettes, MJG01 and MJG02. Diskette MJG01 contained 14 data sets. Diskette MJG02 contained five data sets.

Error Handling during $MIGAID Processing

Errors may occur during $MIGAID processing. The utility handles the following error conditions:

- A disk or diskette unit fails.
- The system fails and you must IPL again.
- You decide to have $MIGAID suspend the operation in progress.
- You specify an invalid volume for the save (SA) or restore (RE).
- You mount the wrong diskette in response to a volume mount request, or you do not have the requested volume.

Example 1 - Disk or Diskette Unit Failure: If the output device fails while writing data, the following appears on your terminal:

```
MIGAWDF - WRITE DATA FAILED FOR FILENAME DISKIO RC=x
          ERROR OCCURRED ACCESSING BLOCKS mmmmm - nnnnn
MIGACFA - COPY OF THIS FILE ABANDONED
```

where x is the READ/WRITE return code and mmmmm - nnnnn is the record range in the data set in which the failure occurred.

Following a disk read failure, $MIGAID attempts to continue saving or restoring the next data set. If $MIGAID encounters a read error on diskette, it attempts error recovery. Error recovery is signaled by the message:

```
  MIGAERS - ERROR RECOVERY PROCEDURE STARTED
```

If the error recovery is successful $MIGAID displays the message:

```
  MIGASER - ERROR RECOVERY SUCCESSFUL
```

and processing continues normally.

If error recovery is not successful, $MIGAID displays:

```
MIGABNR - BLOCK nnnnnn OF dsname1 ON volnm1 NOT RESTORED
          BECAUSE OF READ ERROR AT BLOCK mmmmmm
          OF dsname2 ON volnm2 DISKIO RC= iiiiii
MIGAUER - ERROR RECOVERY UNSUCCESSFUL
```

This error indicates that block nnnnnn on the disk data set (dsname1,volnm1) was not correctly restored because block mmmmmm on the diskette data set (dsname2,volnm2) could not be read. $MIGAID has placed whatever data the diskette unit was able to read from the input block into the output block. Since the read did not complete without error, block mmmmmm of the disk data set may contain incorrect information. You should check carefully any block that $MIGAID was unable to recover. Whenever $MIGAID is not able to recover from a diskette read error, it prompts:

```
  MIGAATF - ABANDON THIS FILE? (Y/N):
```

If you reply 'Y' to this prompt, $MIGAID ceases to process the data set that had the error and starts processing the next one. A reply of 'N' causes $MIGAID to continue processing the file. Any unsuccessful error recoveries are noted in the log. If an I/O error occurs during access to a disk or diskette directory, the following message appears on your terminal:

```
MIGAIED - I/O ERROR READING DIRECTORY ON VOLUME
          DISKIO RC=x
MIGACOA - OPERATION ABANDONED
```

where **x** is the READ/WRITE return code.

Conversion Utilities *(continued)*

This message indicates that $MIGAID has encountered an error so serious that it cannot continue until the problem is corrected. $MIGAID then returns to command input mode. The work file contains the correct status to continue the operation after the problem has been corrected. $MIGAID may be shut down completely (with the EN command) while the corrections are made without loss of restart information.

After correcting a problem, you can resume $MIGAID by entering an * in response to the following prompt:

```
VOLUME TO MIGRATE ("*" FOR CURRENT DEFINITION):
```

or by entering **SA *** to the COMMAND (?): prompt.

Either of these responses directs $MIGAID to use the status in the work file to resume the save process. A similar procedure, using the RE command, can be used to resume an aborted restore process. Note that when the message:

```
MIGACFA - COPY OF THIS FILE ABANDONED
```

is issued, the data set has not been properly saved (or restored) and its contents are undefined. The log will then contain a note that there was an error in the processing of the data set.

Example 2 - System Failure: To recover from a system failure, IPL again, load $MIGAID, and respond:

```
COMMAND (?):  SA *
```

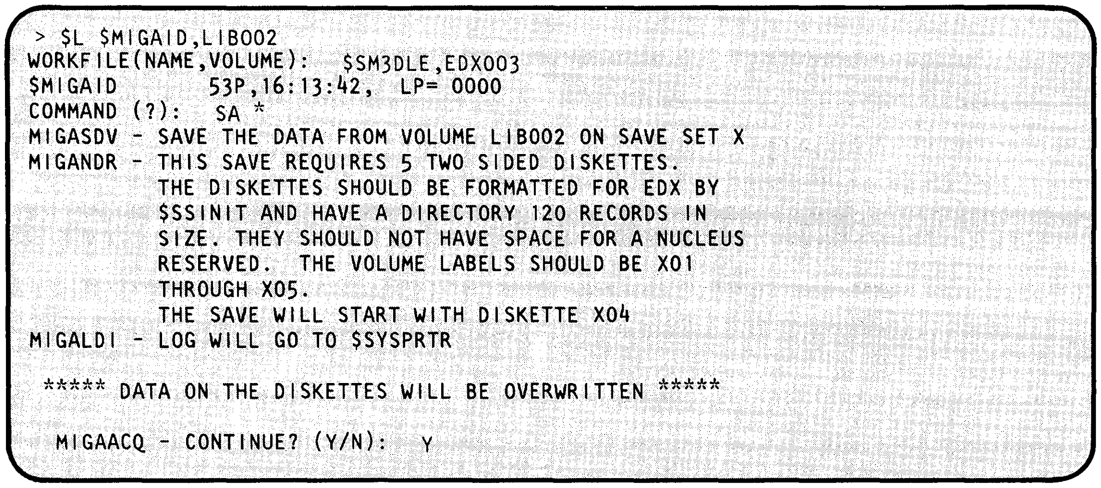
if you are resuming a save or

```
COMMAND (?):  RE *
```

if you are resuming a restore. $MIGAID resumes processing, using its work file status.

In a system failure, the report on the log device may not show as many as three data sets. Use the $DISKUT1 (LA) command to get a complete list of the diskette data sets.

An example of restart:



```
 > $L $MIGAID,LIB002
WORKFILE(NAME,VOLUME):   $SM3DLE,EDX003
$MIGAID        53P,16:13:42,  LP= 0000
COMMAND (?):   SA *
MIGASDV - SAVE THE DATA FROM VOLUME LIB002 ON SAVE SET X
MIGANDR - THIS SAVE REQUIRES 5 TWO SIDED DISKETTES.
          THE DISKETTES SHOULD BE FORMATTED FOR EDX BY
          $SSINIT AND HAVE A DIRECTORY 120 RECORDS IN
          SIZE. THEY SHOULD NOT HAVE SPACE FOR A NUCLEUS
          RESERVED.  THE VOLUME LABELS SHOULD BE X01
          THROUGH X05.
          THE SAVE WILL START WITH DISKETTE X04
MIGALDI - LOG WILL GO TO $SYSPRTR

 ***** DATA ON THE DISKETTES WILL BE OVERWRITTEN *****

  MIGAACQ - CONTINUE? (Y/N):   Y
```

As you can see from this example, $MIGAID will resume processing where it left off.

The log from the restart looks like:

```
MIGASHM - LOG OF SAVE FROM VOLUME LIB002

MIGACDS - DATA SETS COPIED FROM LIB002 TO X04
   DLETEMP3   1802
MIGACDS - DATA SETS COPIED FROM LIB002 TO X05
  DMINTRFS  273  DMINTRFO 60 $MIGAID 64 $MIGAIDO 100
  $MIGAIDL  150  $MIGAIDC  2 TSTUT3  42 $MIGAIDS 679

  MIGADSC - OPERATION SUCCESSFULLY COMPLETED.
```

Conversion Utilities *(continued)*

Example 3 - Suspend an Operation: To cause $MIGAID to suspend an operation in progress, press the attention key (or its equivalent), and enter **STOP**.

Your terminal will look like this after $MIGAID stops:

```
> STOP
MIGAMSD - $MIGAID SHUTTING DOWN SHORTLY
MIGASBU - OPERATION SUSPENDED BY USER
COMMAND (?):
```

$MIGAID completes processing the current data set, prints the log, and issues the command prompt.

In this case, the log and the work file reflect the correct status. Use SA * or RE * to resume, when you are ready.

Example 4 - Invalid Volume Name: If you respond with an invalid name, $MIGAID reissues the prompt:

```
COMMAND (?):  SA XYZZY X
MIGACOL - CAN'T OPEN LIBRARY ON XYZZY
          (LIBRARY NOT FOUND)
COMMAND (?):
```

Example 5 - Wrong Diskette Inserted: If you insert the wrong diskette, $MIGAID reissues the message:

```
MIGAPMV - MOUNT VOLUME volname AND VARY IT ONLINE
          PRESS "ATTN" AND TYPE "GO" WHEN READY
          OR "STOP" TO ABORT MOUNT
```

At this point you may mount the correct diskette and proceed normally. If you do not have the requested diskette, you may abort the mount request by pressing the attention key and typing STOP, as directed. If you choose to abort the mount request, $MIGAID will type the following on your terminal:

```
> STOP
MIGAMSD - $MIGAID SHUTTING DOWN SHORTLY
MIGAVMA - VOLUME MOUNT ABORTED
MIGACOA - OPERATION ABANDONED
COMMAND (?):
```

$MIGRATE

$MIGRATE, part of Version 5, converts $MIGAID output to Version 5 format. If you are not converting to Version 5 from a previous version of EDX, you do not need this program.

Note: $MIGRATE (on EDX Version 5) can be used successfully only if the diskettes being read were created under the following circumstances:

1. The $MIGAID program on the Version 1 or 2 system was obtained from a diskette dated October, 1983 or later. This diskette can be obtained from your IBM Systems Engineer.

2. The EDX system producing the diskettes was at either EDX Version 1 Modification Level 3 PTF P0A or EDX Version 2 Modification Level 1 PTF P07.

$MIGRATE uses a work data set which you must allocate with $DISKUT1 and clear with $DISKUT2 before you invoke $MIGRATE for the first time. The $MIGRATE work data set must be at least 121 records in size. When converting a $MIGAID save set, $MIGRATE keeps checkpoint and other internal information in its work file, allowing $MIGRATE to recover from power or other system failure. Do not delete or clear a $MIGRATE work data set until $MIGRATE has successfully completed a restore operation. Do not allocate the $MIGRATE work file on the disk volume that is being restored (the target volume) since, if there is a data set of the same name on the save set, the work file would be replaced with the saved data.

$MIGRATE Commands

$MIGRATE accepts the following commands:

?	Help. Prints the following list of commands and explanations.
EN	End. Command to end $MIGRATE.
PR	Printer. Names the logical printer to which logs should go. 'PR *' means the current terminal. Default: PR $SYSPRTR
RE	Restore. Restores all data members saved with the Version 1 or 2 utility $MIGAID to a Version 5 disk.
RG	Restore generic. Similar to RE but restores only data members having names starting with the generic text you supply.
RN	Restore nongeneric. Similar to RG but restores only data members having names that do not start with the generic text you supply.

In addition $MIGRATE has the following attention commands:

STOP	Stops a restore that is in process.
GO	Causes $MIGRATE to start again when it has paused for more diskettes.

$MIGRATE is much like the restore portion of its Version 1 and 2 counterpart, $MIGAID, with the addition of the ability to restore only those data sets that match (or do not match) a given generic string. Error recovery procedures and error message sequences for $MIGRATE are identical to those for $MIGAID.

$MIGCOPY

The $MIGCOPY program copies data sets on diskette in Version 1 or 2 format to either disk or diskette in Version 5 format. If you do not have any Version 1 or 2 format diskettes, you do not need this program.

$MIGCOPY operates very much like $COPYUT1 with the following exceptions:

- Its input (source) data set format is Version 1 or 2.
- Its output (target) data set format is Version 5.
- It does not copy program-type data sets (Version 1 or 2 programs will not run on Version 5 unless they are recompiled).
- It has a $DISKUT1-like "LA" command to list the contents of the Version 1 or 2 source diskette.

$MIGCOPY Commands

$MIGCOPY accepts the following commands:

?	Help. Prints the following list of commands and their explanations.
EN	End. Command to end $MIGCOPY.
CM	Copy Member. Copies a specific data member from the Version 1 or 2 source volume to the Version 5 target volume.
CALL	Copy All. Copies all data members from the Version 1 or 2 source volume to the Version 5 target volume.
CG	Copy Generic. Similar to CALL but copies only data members having names starting with the generic text you supply.
CNG	Copy nongeneric. Similar to CG but copies only data members having names that do not start with the generic text you supply.

CV	Change volume. Changes the source and target volumes that will be used.
LA	Lists all members on source volume.
SQ	Turns question mode on for all multiple copy operations.
NQ	Turns question mode off. (This is the default.)

In addition, $MIGCOPY has the following attention commands:

STOP	Stop an in-process multiple copy.
GO	Cause $MIGCOPY to start again when it has paused for a volume mount.

STOP and GO are entered to get $MIGCOPY's attention when it is doing a multiple copy operation or when it has paused for you to mount a diskette. To enter an attention command, press the attention key (or its equivalent). The system responds with ">" indicating its readiness to accept an attention command. Type the command and press the enter key.

During a multiple copy command, the STOP command causes $MIGCOPY to stop after it has finished processing the data set it is copying.

If $MIGCOPY needs a diskette that is not mounted, it pauses and asks you to mount it. The GO attention command is used to inform $MIGCOPY that the requested diskette is mounted.

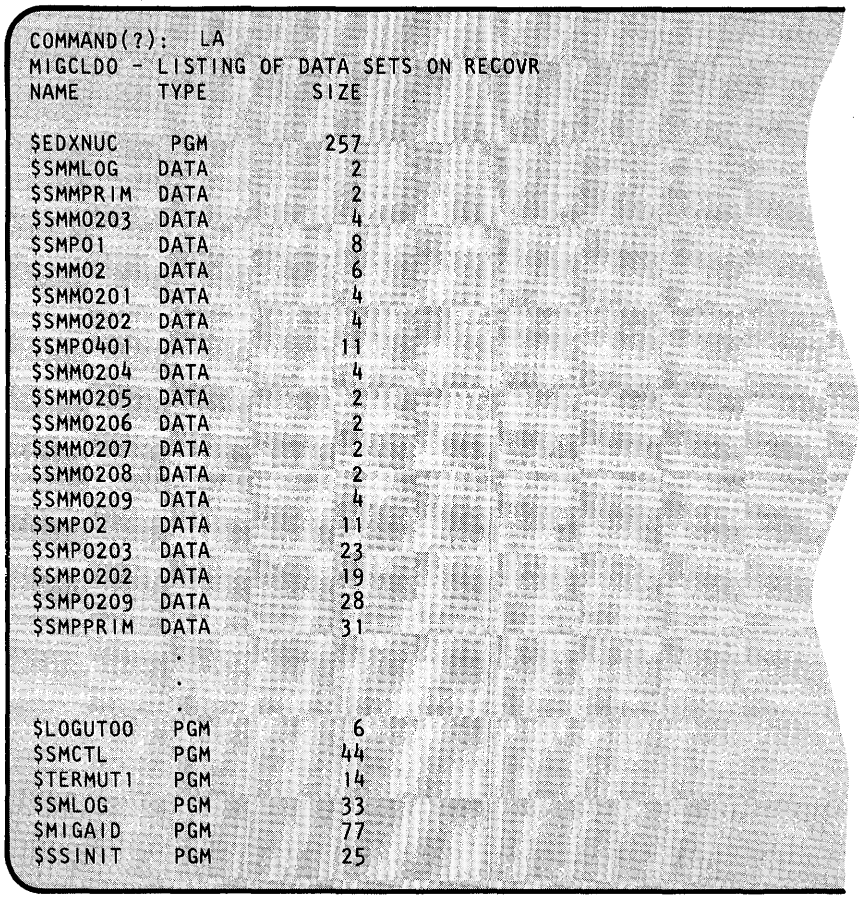
Example 1 - Loading $MIGCOPY: After $MIGCOPY is loaded, it prompts you for the names of the source (input) diskette volume and the target (output) volume. The source volume must be in Version 1 or 2 format and the target in Version 5 format. For example:

```
> $L $MIGCOPY
$MIGCOPY 50P,01:00:08, LP= 0000
SOURCE VOLUME:  RECOVR
TARGET VOLUME:  EDX005
```

The source and target volumes may be changed later by using the CV command.

Conversion Utilities *(continued)*

Example 2 - Using the List All (LA) Command: The LA command may be used to list the names of the data sets on the source volume. $DISKUT1, which is usually used to list volume contents, executes only on Version 5 format diskettes. To list Version 1 or 2 diskettes, use the $MIGCOPY LA command. For example:



```
COMMAND(?):  LA
MIGCLDO - LISTING OF DATA SETS ON RECOVR
NAME       TYPE        SIZE

$EDXNUC     PGM         257
$SMMLOG    DATA           2
$SMMPRIM   DATA           2
$SMM0203   DATA           4
$SMP01     DATA           8
$SMM02     DATA           6
$SMM0201   DATA           4
$SMM0202   DATA           4
$SMP0401   DATA          11
$SMM0204   DATA           4
$SMM0205   DATA           2
$SMM0206   DATA           2
$SMM0207   DATA           2
$SMM0208   DATA           2
$SMM0209   DATA           4
$SMP02     DATA          11
$SMP0203   DATA          23
$SMP0202   DATA          19
$SMP0209   DATA          28
$SMPPRIM   DATA          31
             .
             .
             .
$LOGUT00    PGM           6
$SMCTL      PGM          44
$TERMUT1    PGM          14
$SMLOG      PGM          33
$MIGAID     PGM          77
$SSINIT     PGM          25
```

Note that although the LA command lists all data sets on the diskette, the copy commands will copy only data-type members.

Example 3 - Using the Copy Commands: The CM command copies a single data set from the source volume to the target volume. For example:

```
COMMAND(?):  CM
SOURCE (FROM) MEMBER NAME:  CALCSRC
TARGET (TO) MEMBER NAME
   (ENTER * FOR SAME NAME AS SOURCE MEMBER):  *

MIGCHRC - MEMBER CALCSRC COPIED.    33 BLOCKS PROCESSED
```

Like all $MIGCOPY copy commands, the CM command automatically allocates the target data set. If a data set of the same name already exists on the target volume, it will be deleted before the allocation takes place.

The CALL command copies all data-type data sets from the source volume to the target volume. The data sets will have the same names on the target that they had on the source. For example:

```
COMMAND(?):  CALL
MIGCHRC - MEMBER $SMMLOG  COPIED.     2 BLOCKS PROCESSED
MIGCHRC - MEMBER $SMMPRIM COPIED.     2 BLOCKS PROCESSED
MIGCHRC - MEMBER $SMM0203 COPIED.     4 BLOCKS PROCESSED
MIGCHRC - MEMBER $SMP01   COPIED.     8 BLOCKS PROCESSED
                 .
                 .
                 .
```

The CG and CNG commands, like the CALL command, copies multiple data sets with a single command, but they each have an additional parameter, the "generic text" that allows selective copying. Before each data set is copied, its name is compared with the generic text you supply. If the name begins with the same characters as the generic text, the CG command copies that data set. The CNG command works in just the opposite way. If the name does *not* match the generic text, the data set will be copied.

For example, to copy all data sets whose name begins with the characters "$4":

```
COMMAND(?):  CG $4
MIGCHRC - MEMBER $4978IS0 COPIED.    8 BLOCKS PROCESSED
MIGCHRC - MEMBER $4978CS0 COPIED.   16 BLOCKS PROCESSED
MIGCHRC - MEMBER $4978CS1 COPIED.   16 BLOCKS PROCESSED
MIGCHRC - MEMBER $4978IS1 COPIED.    8 BLOCKS PROCESSED
```

Conversion Utilities *(continued)*

The following example copies all data sets whose names do not begin with the character "$":

```
COMMAND(?):  CNG $
MIGCHRC - MEMBER CALCSRC COPIED.    33 BLOCKS PROCESSED
```

Only one data set is copied - CALCSRC.

Example 4 - Controlling Prompting: Besides the ability to selectively copy using generic text match, the commands SQ and NQ allow you to turn on or off a mode in which you are prompted before each data set is copied. At that time you tell $MIGCOPY whether or not to copy that particular data set. The SQ command turns the mode on; the NQ command turns if off. If on, it is effective for all multiple data set copy commands:

```
COMMAND(?):  SQ
COMMAND(?):  CG $
RESTORE $SMMLOG ? (Y/N):  N
RESTORE $SMMPRIM? (Y/N):  N
RESTORE $SMM0203? (Y/N):  N
RESTORE $SMP01  ? (Y/N):  Y
MIGCHRC - MEMBER $SMP01    COPIED.     8 BLOCKS PROCESSED
RESTORE $SMM02  ? (Y/N):  Y
MIGCHRC - MEMBER $SMP02    COPIED.     6 BLOCKS PROCESSED
RESTORE $SMM0201? (Y/N):  N
RESTORE $SMM0202? (Y/N):  N
              .
              .
              .
```

Example 5 - Using the Change Volume (CV) Command: The CV command allows you to change the source and target volume assignments. For example, to change the target volume to EDX002:

```
COMMAND(?):  CV
MIGCSVI - SOURCE VOLUME IS RECOVR OK? (Y/N):  Y
MIGCTVI - TARGET VOLUME IS EDX005 OK? (Y/N):  N EDX002
```

Appendix A. System Definition Statements

This appendix describes the definition statements used to define to your supervisor the I/O devices attached to your Series/1. The following definition statements are used to describe your system:

- ADAPTER - Defines a multiline attachment
- BSCLINE - Defines a binary synchronous line
- DISK - Defines direct access storage devices
- EXIODEV - Defines EXIO interface devices
- HOSTCOMM - Defines host communication support
- SENSORIO - Defines sensor I/O devices
- SYSTEM - Defines processor characteristics
- TAPE - Defines tape device
- TERMINAL - Defines terminals
- TIMER - Defines system timer feature

These statements are used in the system generation process only which is described in Chapter 5, "Generate a Tailored Operating System" on page IS-77.

For an explanation of the format and syntax rules for coding definition statements, refer to the *Language Reference.*

ADAPTER - Define a Multiline Attachment

ADAPTER defines the following attachments to be supported in your generated system:

- Multifunction Attachment Feature #1310 (MFA)
- Printer Attachment - 5200 Series Feature #5640 (ALPA)
- Multidrop Work Station Attachment Feature #1250 (SMIO)

One ADAPTER statement is required for each attachment. All ADAPTER statements must be grouped together and must precede the definition of the first device attached to a specific attachment. The last ADAPTER statement specified must include an END=YES specification.

For the Multifunction Attachment (MFA), each ADAPTER statement provides the attachment base address (the lowest device address in the attachment's range of up to four device addresses) and the names of the devices attached to the system through the attachment.

For the Printer Attachment - 5200 Series (ALPA), each ADAPTER statement provides the attachment base address (the lowest device address in the attachment's range of up to eight device addresses) and the names of the devices attached to the system through the attachment. You can attach up to eight printers through the ALPA attachment.

For the Multidrop Work Station Attachment (SMIO), each ADAPTER statement provides the attachment base address (the lowest address in the attachment's range of up to eight device addresses) and the names of the devices attached to the system through the attachment. You can attach up to eight 4980 display stations through the SMIO attachment.

Notes:

1. Each device to be connected through the MFA should be defined exclusively with either the TERMINAL and BSCLINE statements or the EXIODEV statement. A combination of system-supported and user-supported devices connected through a common attachment may produce unpredictable results.

2. Each device you are connecting through the ALPA and SMIO attachments must be defined by a separate TERMINAL statement.

3. Modem support for MFA (1310) is half-duplex only.

ADAPTER - Define a Multiline Attachment *(continued)*

Syntax:

```
label        ADAPTER     ADDRESS=,TYPE=,DEVICES=(list),END=

Required:    label,ADDRESS=,TYPE=,DEVICES=
Default:     END=NO
```

Operand	***Description***
label	A 1 to 8 alphanumeric character label beginning with a letter or one of the following special characters: $, #, or @. Required for each ADAPTER statement.
ADDRESS=	The base address (two hexadecimal digits) of the attachment. For the MFA, this address must be divisible by four. For the ALPA and SMIO, this address must be divisible by eight.
TYPE=	One of the following: **MFA** Defines the Multifunction Attachment (#1310) **ALPA** Defines the Printer Attachment - 5200 Series (#5640) **SMIO** Defines the Multidrop Work Station Attachment (#1250).
DEVICES=	A list of one to eight labels depending on the type of of attachment: • For an MFA, a list of one to four labels on the BSCLINE and TERMINAL statements that define the devices attached to the attachment. • For an ALPA attachment, a list of one to eight labels on the TERMINAL statement that defines the devices attached to the attachment. • For an SMIO attachment, a list of one to eight labels on the TERMINAL statement that defines the devices attached to the attachment.
END=	YES, for the last ADAPTER statement in the system definition module. The default is END=NO.

ADAPTER - Define a Multiline Attachment *(continued)*

Example 1: A Multifunction Attachment with four devices attached:

- One BSCLINE at address 58
- Two 3101 Model 23s in block mode at addresses 59 and 5A
- One 4975-02L printer at address 5B

```
MFA01     ADAPTER    ADDRESS=58,TYPE=MFA,                               C
                DEVICES=(BSC1,$SYSLOG,$SYSLOGA,$SYSPRTR),END=YES

BSC1      BSCLINE    ADDRESS=58,ADAPTER=MFA,END=YES

$SYSLOG   TERMINAL   DEVICE=ACCA,ADAPTER=MFA,ADDRESS=59,                C
                MODE=3101B,LMODE=RS422

$SYSLOGA TERMINAL   DEVICE=ACCA,ADAPTER=MFA,ADDRESS=5A,                C
                MODE=3101B,LMODE=RS422

$SYSPRTR TERMINAL   DEVICE=4975-02L,ADAPTER=MFA,ADDRESS=5B,            C
                END=YES
```

Example 2: A Printer Attachment - 5200 Series (ALPA) at address 58 with two devices attached.

- One 5219 printer at address 58
- One 5219 printer at address 59

Note: In this example, the address of the attachment and the first device are the same. This is valid.

```
ALPA01    ADAPTER    ADDRESS=58,TYPE=ALPA,                              C
                DEVICES=($SYSLOG,$SYSLOGA),END=YES

$SYSLOG   TERMINAL   DEVICE=5219,ADAPTER=ALPA,ADDRESS=58,               C
                SECADDR=01,PORT=0

$SYSLOGA TERMINAL   DEVICE=5219,ADAPTER=ALPA,ADDRESS=59,               C
                SECADDR=02,PORT=0
```

ADAPTER - Define a Multiline Attachment *(continued)*

Example 3: A Multidrop Work Station Attachment (SMIO) at address 80 with two devices attached.

- One 4980 display station at address 80
- One 4980 display station at address 81

Note: In this example, the address of the attachment and the first device are the same. This is valid.

```
SMIO01     ADAPTER    ADDRESS=80,TYPE=SMIO,                              C
                 DEVICES=(TERM1,TERM2),END=YES

TERM1      TERMINAL   DEVICE=4980,ADAPTER=SMIO,ADDRESS=80,               C
                 SECADDR=01,PORT=0

TERM2      TERMINAL   DEVICE=4980,ADAPTER=SMIO,ADDRESS=81,               C
                 SECADDR=02,PORT=0
```

BSCLINE - Define a Binary Synchronous Line

BSCLINE defines a binary synchronous line to be supported in the generated system. One BSCLINE statement is required for each line to be referenced by programs using the Binary Synchronous Communications Access Method. (Refer to the *Communications Guide* for a description of the Binary Synchronous Communications Access Method.)

Notes:

1. Code a BSCLINE statement if you use the Remote Management Utility or the X.21 Circuit Switched Network support.

2. Group all BSCLINE statements together with the last BSCLINE statement, including an END=YES specification.

Syntax:

```
label       BSCLINE ADDRESS=,TYPE=,RETRIES=,MC=,ADAPTER=,
            POLL=(list),END=

Required:   label if ADAPTER=MFA
Defaults:   ADDRESS=09,TYPE=PT,RETRIES=6,MC=NO,END=NO
```

Operand	*Description*
label	A 1 to 8 alphanumeric character label beginning with a letter or one of the following special characters: $, #, or @. Optional unless ADAPTER=MFA. If ADAPTER=MFA, this label must correspond to the label in the DEVICE= list on the ADAPTER definition statement.
ADDRESS=	The hardware address (in hexadecimal) of the line. The default is ADDRESS=09. For the X21 Circuit Switched Network support, you must specify an even address.
TYPE=	PT (the default) — The line is a point-to-point, nonswitched line with a single remote station. The adapter should be jumpered with DTR permanently enabled. The MFA does not have the DTR jumper. **Note:** Do not use the PT option with the X.21 Circuit Switched Network.

BSCLINE - Define a Binary Synchronous Line *(continued)*

AC (auto-call) — The line is a point-to-point, switched line. The IBM 2080 Synchronous Communication Single-Line Control/High Speed Feature should be jumpered for switched line operation and binary synchronous communications. Auto call is established through the EDX X.21 support. For information on using the auto-call option, see the *Communications Guide*.

DC (direct-call) — The line is predefined direct on a point-to-point, switched line. The IBM 2080 Synchronous Communication Single-Line Control/High Speed Feature should be jumpered for switched line operation and binary synchronous communications. Direct call is established through the EDX X.21 support. For information on using the direct-call option, see the *Communications Guide*.

SM — The line is on a switched network and connection will be established manually by the operator. The adapter should be jumpered for switched line operation and DTR should *not* be permanently enabled. The MFA does not have the switched line operation and DTR jumpers.

Note: If you use the SM option with the X.21 Circuit Switched Network, it defaults to auto call.

SA — The line is on a switched network and calls should be answered automatically by the BSC Access Method (during BSCOPEN). The adapter should be jumpered for switched line operation and DTR should not be permanently enabled. The MFA does not have the switched line operation and DTR jumpers.

Note: If you use the SA option with the X.21 Circuit Switched Network, it defaults to auto answer.

MC — The Series/1 is the controlling station on a multipoint line. The adapter should be jumpered with DTR permanently enabled and multipoint line should not be jumpered. The MFA does not have the DTR and multipoint line jumpers.

Note: Do not use the MC option with the X.21 Circuit Switched Network.

BSCLINE - Define a Binary Synchronous Line *(continued)*

MT — The Series/1 is a tributary station on a multipoint line. The adapter should be jumpered for multipoint tributary operation with DTR permanently enabled. The MFA does have the multipoint tributary operation jumpers; it does not have the DTR jumpers. However, the physical jumpers are used only prior to initialization time. The logical jumpering and random access memory load override the physical jumpering after initialization.

Note: Do not use the MT option with the X.21 Circuit Switched Network.

RETRIES= The number of attempts which should be made to recover from common error conditions before posting a permanent error. The default is RETRIES=6.

MC= NO (the default) — The binary synchronous adapter located at the address specified in the ADDRESS= operand is one of the following: a medium speed, single-line feature card; a high speed, single-line feature card; an X.21 card jumpered to a run as a leased Bisync card; or an MFA (#1310).

YES — The binary synchronous adapter located at the address specified in the ADDRESS= operand is part of a multiline controller feature configuration. When generating supervisors using multiline controller attachments, note the following:

- The character string YES must be specified. Any other character string will be equivalent to NO.
- All multiline feature cards must start at a base address ending with either X'0' or X'8'. A BSCLINE statement must exist for the line at this base address if any of the other lines of the multiline attachment are to be used.

Note: Do not use the MC operand with the X.21 Circuit Switched Network.

ADAPTER= MFA — the line is on a Multifunction Attachment (MFA)

Notes:

1. If a bisync line is used with the MFA, it must be on the MFA base address.
2. Do not use the ADAPTER operand with the X.21 Circuit Switched Network.

BSCLINE - Define a Binary Synchronous Line *(continued)*

POLL= A list of 1 to 4 poll/select addresses (two digit hexadecimal) to which the line should respond. Applies only if ADAPTER=MFA and TYPE=MT.

Note: Do not use the POLL operand with the X.21 Circuit Switched Network.

END= YES, for the last BSCLINE statement in the system definition module. The default is END=NO.

Example 1: A point-to-point, nonswitched binary synchronous line located at address 28 that is not part of a multiline controller configuration.

```
         BSCLINE ADDRESS=28,TYPE=PT,RETRIES=10,MC=NO
```

Example 2: A multipoint binary synchronous line at address 58, attached to a Series/1 through an MFA. A list of three poll/select addresses is specified.

```
BSC1     BSCLINE ADDRESS=58,ADAPTER=MFA,TYPE=MT,POLL=(C0,C1,C2)
```

Example 3: A binary synchronous line at address 30, on a switched network. An operator will establish this line connection manually.

```
         BSCLINE ADDRESS=30,TYPE=SM,RETRIES=2,MC=YES,END=YES
```

Example 4: A point-to-point binary synchronous line at address 30, on a switched network.

```
         BSCLINE ADDRESS=30,TYPE=AC,RETRIES=2,END=YES
```

DISK - Define Direct Access Storage

DISK defines the direct access storage devices to be supported in the generated system. All DISK statements must be grouped together with the TAPE statements. The last DISK OR TAPE statement must include an END=YES specification.

Note: We recommend placing the DISK statements that represent disk devices in front of those representing diskette devices. This ensures that disks are still accessible if a defective diskette is encountered. If you define all your disks before your diskettes, you will get better performance.

Syntax:

```
blank      DISK         DEVICE=,ADDRESS=,VOLNAME=(namelist),
                        TASK=,END=

Required:  DEVICE=,ADDRESS=
Defaults:  END=NO,TASK=NO
```

Operand	*Description*
DEVICE=	One of the following device types:

Type	**Description**
4964	4964 Diskette Unit
4965	4965 Diskette Unit within the 4952 Model C or 30D processor, the 4954 or 4956 Model 30D or 60D processors, or the 4965 Storage and I/O Expansion Unit
4966	4966 Diskette Magazine Unit
4962-1	4962-1 Disk (9.3-megabyte unit)
4962-1F	4962-1F Disk (9.3-megabyte unit with fixed heads)
4962-2	4962-2 Disk (9.3-megabyte unit with a diskette unit)
4962-2F	4962-2F Disk (9.3-megabyte unit with fixed heads and a diskette unit)
4962-3	4962-3 Disk (13.9-megabyte unit)
4962-4	4962-4 Disk (13.9-megabyte unit with a diskette unit)
4963-29	4963-29 Disk (29-megabyte unit)
4963-23	4963-23 Disk (23-megabyte unit with fixed heads)

DISK - Define Direct Access Storage *(continued)*

4963-64	4963-64 Disk (64-megabyte unit)
4963-58	4963-58 Disk (58-megabyte unit with fixed heads)
4967-2	4967-2 Disk (one or two 200 megabyte units)
4967-4	4967-4 Disk (one to four 200 megabyte units)
DDSK-30	30-megabyte Disk within the 4952, 4954, or 4956 Model 30D processors or the 4965 storage and I/O expansion unit (model 30D)
DDSK-60	60-megabyte disk within the 4954 or 4956 Model 60D processors or the 4965 storage and I/O expansion unit (model 60D)

ADDRESS= The hexadecimal address of the unit.

Notes:

1. If you have a 4965 diskette unit within the 4952 Model 30D processor,the 4954 or 4956 Model 30D/60D processors, or within the 4965 storage and I/O expansion unit, use a separate DISK statement to define the unit. The address of the diskette unit must be one greater than the DDSK-30 or DDSK-60. For example, if the disk is defined at address X'44', then the diskette unit must be address X'45'.

2. If you have a 4965 diskette unit, use a separate DISK statement for each diskette slot within the unit. The address of the second diskette slot must equal the address of the first diskette slot plus 1.

VOLNAME= A list of volume names (1 to 6 characters) that will be located on the device. These volumes are designated as performance volumes. The system finds the disk addresses of the specified volumes at IPL time.

See "Disk Considerations:" on page IS-47 for an explanation of performance volumes.

Each volume you specify requires 46 bytes of storage.

Notes:

a. You cannot designate performance volumes on a diskette. As a result, this operand is ignored if DEVICE=4964, 4965, or 4966.

b. This parameter is optional; you must allocate the volumes with the $INITDSK utility.

c. Volume names, tape IDS, and TERMINAL statement labels must be unique. No tape ID or TERMINAL statement label can match a volume name.

DISK - Define Direct Access Storage *(continued)*

TASK= NO, or omit, if a new task is not required (the default). An I/O task is automatically generated; specify NO to prevent generation of an additional task.

YES, to cause the system to generate a new I/O task. This task is used to service I/O requests for this and subsequent devices until the system encounters a new DISK statement with TASK=YES.

Specifying TASK=YES on a DISK statement allocates a task control block that is used in servicing READ and WRITE requests for the group of devices being defined. The effect is to allow READ and WRITE requests to proceed in parallel with requests to other groups of devices. With the exception of the 4963 disk, the resulting overlap may improve performance significantly when concurrent requests to different groups of devices occur. To achieve maximum flexibility and performance, you should specify TASK=YES on each DISK statement. Each TASK=YES requires 128 bytes of additional storage.

END= YES, for the last DISK or TAPE statement in the system definition module. The default is END=NO.

Example 1: One I/O task that is shared by all direct access drives.

```
DISK   DEVICE=4962-1F,ADDRESS=03
DISK   DEVICE=4963-23,VOLNAME=(EDX002,ASMLIB),ADDRESS=48
DISK   DEVICE=4964,ADDRESS=02
DISK   DEVICE=4965,ADDRESS=44,END=YES
```

Note: END=YES is required only once for the DISK/TAPE definition statements.

Example 2: Each disk has an I/O task. One I/O task for the two 4964s and a second I/O task for the 4962.

```
DISK   DEVICE=4964,ADDRESS=02,TASK=YES
DISK   DEVICE=4964,ADDRESS=12,TASK=YES
DISK   DEVICE=4962-1F,ADDRESS=03,TASK=YES,END=YES
```

Example 3: DISK definition statement defining the 60-megabyte disk and an optional 4965 diskette unit.

```
DISK   DEVICE=DDSK-60,ADDRESS=44
DISK   DEVICE=4965,ADDRESS=45,END=YES
```

EXIODEV - Define EXIO Interface Device

EXIODEV defines the devices to be supported via the EXIO interface in the generated system. All EXIODEV statements must be grouped together. The last EXIODEV statement must include an END=YES specification.

An EXIODEV statement must be defined for each:

- System/370 Channel Device attached to your system
- Physical unit that defines an SDLC line for the Series/1 Systems Network Architecture.

For more information on defining channel attach devices, refer to the Program Directory for the Series/1-System/370 Channel Attach (5719-CX1). For more information on defining an SDLC line, refer to the Program Directory for the Series/1 Systems Network Architecture (5719-SX1).

Notes:

1. Any device defined via EXIODEV should not be defined on any other statement such as DISK or BSCLINE. Doubly-defined devices will cause unpredictable results when accessed by a combination of READ/WRITE and EXIO. The system does not initialize any device you define with EXIODEV. You must load any control store that is required by the device.

2. If you define devices connected through a multiple-device feature (controller), the devices should be either defined exclusively with the EXIODEV statement or the TERMINAL or BSCLINE statements. A combination of system supported and user supported devices connected through a common attachment may produce unpredictable device.

Syntax:

```
blank        EXIODEV       ADDRESS=,END=,MAXDCB=,RSB=

Required:    ADDRESS=
Defaults:    MAXDCB=0,RSB=0,END=NO
```

Operand	*Description*
ADDRESS=	The device address (two hexadecimal digits). For a system with SNA support, this address must match the address specified for the ADDRESS= operand of the SNAPU definition statement.
MAXDCB=	The maximum number of chained DCBs which will be used for this device. Must be eight or less. The default is MAXDCB=0. **Note:** If the device you define is an OEM or IBM device (nonstandard support), refer to the device description manual for that device for MAXDCB count.

EXIODEV - Define EXIO Interface Device *(continued)*

For a system with SNA support, the value specified must be 2–8. This value must be equal to the value specified on the DCBNO= operand on the SNAPU definition statement and one greater than the value specified for the MAXOUT parameter of the network control program (NCP).

For a system with one System/370 Channel Attach device, specify MAXDCB=2.

For a system with more than one System/370 Channel Attach device, specify MAXDCB=2 on one EXIODEV statement and MAXDCB=1 on the others.

For a system with a Data Collection Interactive RPQ, specify MAXDCB=1.

RSB= The number of residual status bytes the device will transfer. Enter zero or an even decimal number between 4 and 16 inclusive. The default is RSB=0.

Note: If the device you define is an OEM or IBM device (nonstandard support), refer to the device description manual for that device for RSB count.

For a system with SNA support, specify RSB=4.

For a System/370 Channel Attach device, specify RSB=6.

For a system with a Data Collection Interactive RPQ, specify RSB=4.

END= Specify YES for the last EXIODEV statement in the system definition module. The default is END=NO.

Examples:

```
EXIODEV   ADDRESS=00

EXIODEV   ADDRESS=E0,RSB=12,MAXDCB=2

EXIODEV   ADDRESS=10,RSB=6,MAXDCB=2

EXIODEV   ADDRESS=11,RSB=6,MAXDCB=1,END=YES
```

HOSTCOMM - Define Host Communications Support

HOSTCOMM defines the device type and address to be used for host communication support in the generated system. This support operates with the Host Communications Facility Installed User Program (IUP).

Syntax:

```
blank       HOSTCOMM   DEVICE=,ADDRESS=

Required:   DEVICE=,ADDRESS=
Defaults:   None
```

Operand	*Description*
DEVICE=	BSCA, for Binary Synchronous Communications Adapter support. This is the only device supported and must be a single line BSC adapter (feature 2074 or 2075). Only one is allowed.
ADDRESS=	The hexadecimal address of the adapter.

Example:

```
          HOSTCOMM  DEVICE=BSCA,ADDRESS=09
```

SENSORIO - Define Sensor I/O Devices

SENSORIO defines the sensor I/O devices to be supported in the generated system. All SENSORIO statements must be grouped together with the last one including an END=YES specification.

Syntax:

```
blank        SENSORIO      ADDRESS=,DEVICE=,AI=,AO=,DI=,DO=,PI=,AITYPE=,
                           LEVEL=,END=

Required:    DEVICE=,ADDRESS=
Defaults:    AITYPE=RELAY,LEVEL=1,END=NO
```

Operand	***Description***
ADDRESS=	The base address of the device (in hexadecimal). This is the only required address if DEVICE=IDIO unless PI is needed on this unit.
DEVICE=	One of the following device types:

IDIO	The integrated digital I/O nonisolated feature (feature #1560)
4982	The sensor I/O unit

AI=	The address or list of addresses of the analog input multiplexor feature(s) on this device.
AO=	The address or list of addresses of the analog output point(s) on this device.
DI=	The address or list of addresses of the digital input group(s) on this device.
DO=	The address or list of addresses of the digital output group(s) on this device. PI can be read as DI.
PI=	The address or list of addresses of the digital input group(s) to be used as process interrupt.

Note: For the AI, AO, DI, DO and PI operands, multiple addresses must be included in parentheses.

SENSORIO - Define Sensor I/O Devices *(continued)*

AITYPE=	The type of AI multiplexer(s). Valid entries are: • RR or RELAY - for relay (the default) • SS or SOLID - for solid state **Note:** The names have a one-to-one relationship with addresses on the AI operand. Multiple entries must be included in parentheses.
LEVEL=	A number (from 0-3) to assign the hardware interrupt level to the device. The default is LEVEL=1. **Note:** This assignment is for all features on that device.
END=	YES, for the last SENSORIO statement in the system definition module. The default is END=NO.

Examples:

```
SENSORIO  DEVICE=IDIO,ADDRESS=68

SENSORIO  DEVICE=4982,ADDRESS=60,AO=65,DO=62,DI=64,      C
      PI=63,AI=61,AITYPE=SS

SENSORIO  DEVICE=4982,ADDRESS=70,DI=(70,71)

SENSORIO  DEVICE=4982,ADDRESS=60,AI=(62,63),             C
      AITYPE=(RELAY,SOLID),AO=64,DI=(65,66),DO=67

SENSORIO  DEVICE=IDIO,ADDRESS=68,PI=68,END=YES
```

SYSTEM - Define Processor Storage

SYSTEM defines the characteristics of the processor storage, including the partition structure, the storage assigned to each partition for execution of application programs, and the number of programs that can concurrently execute in each partition. This statement must be specified once.

Note: You need only define the number of partitions required for program execution. If you generate a multipartition supervisor and place supervisor code in a partition that you have not defined on the SYSTEM statement, the partition is automatically defined. However, this partition will only contain supervisor code and cannot be used for execution of application programs.

The Event Driven Executive supports two types of storage: mapped and unmapped. Mapped storage is the storage you define with the PART= operand. Unmapped storage is any storage not defined with the PART= operand or any storage above 512K. The system acquires unmapped storage through the unmapped storage instructions. See the *Language Reference* for a description of these instructions. For an explanation of using unmapped storage for application programs, see the *Event Driven Executive Language Programming Guide.*

Syntax:

```
blank       SYSTEM      MAXPROG=,PARTS=,DATEFMT=,IABUF=,
                        XPSSTK=,COMMON=,INITPRT=,INITMOD=,MECBLST=

Required:   None
Defaults:   MAXPROG=10,PARTS=32,DATEFMT=MMDDYY
            IABUF=20,XPSSTK=20,COMMON=EDXSYS
```

Operand	***Description***
MAXPROG=	The maximum number of concurrently executing programs to be allowed in a partition. Add 1 to your calculated number for each occurrence of $JOBUTIL in that partition. Add 2 for each occurrence of the session manager in that partition. Four words of storage are required in the nucleus for each program specified. The default is MAXPROG=10.
	You must specify a list of the maximum number of concurrently executing programs allowed in each partition. The number of programs that can run concurrently in a system varies with each installation depending on the size of your processor storage and programs and your processor time requirements.
	If your system has unmapped storage, you need to specify a greater number of programs because unmapped storage generally uses more storage within a partition. For example, if you would normally define five (5) programs, specify seven (7) instead.

SYSTEM - Define Processor Storage *(continued)*

PARTS= The number of 2K (1K=1024 bytes) blocks of storage to be assigned to each partition for execution of application programs. You can specify up to 32 2K blocks of storage or up to a total of 64K for each partition. Depending on the size of your supervisor, you may not receive the amount of user space anticipated because of the amount of storage taken up by supervisor code within a partition. Enter a list showing the maximum size of each partition. You can define up to eight 64K partitions for the 4955 and 4956, up to four 64K partitions for the 4954, and up to two 64K partitions for the 4952. The list must contain the same number of entries as the list coded for MAXPROG=. The default is PARTS=32.

The user partitions (1–8) must fit in corresponding address spaces or map areas that are respectively numbered 0–7 and can contain 64KB of designated storage.

The amount of storage available in any partition is the smaller of:

- The size you define in the PARTS= parameter
- 64K minus the size of the supervisor.

(Refer to Chapter 4, "Select Your Required Support" on page IS-39 for information on how to estimate supervisor size.)

The maximum value that can be specified is 32; the minimum is 0. When you specify the size to be assigned to partition 1 and you wish partition 1 to have all storage not used by the supervisor, code 32 rather than calculating the value. Otherwise, you must calculate the size of partition 1.

To define unmapped storage, specify less than 32 2K blocks of storage for each partition. For example, if you have three partitions in a 192K processor and only define PARTS=(32,16,16), you only map 128K of processor storage. The balance of 32 2K blocks is unmapped storage.

(To determine the storage required by licensed programs such as the Indexed Access Method and the Multiple Terminal Manager, refer to work sheet 4 in Appendix E.)

SYSTEM - Define Processor Storage *(continued)*

DATEFMT= The format to be used when the date is displayed (PRINDATE or $W) or when entering the date via $T. A return code is set in response to a GETTIME request with the DATE option.

Specify MMDDYY for a date format of month.day.year. Specify DDMMYY for a date format of day.month.year. MMDDYY is the default.

Note: You must include timer support in your supervisor to have date support.

IABUF= The maximum number of interrupts that may be buffered by the task supervisor. You must code a value between 10 and 100. The default is 20 and is adequate for most systems. The value should be increased if the system could be overloaded by a large number of interrupts. (The system will stop or enter a continuous run loop.) Each increment increases the supervisor storage requirements by eight bytes.

Note: Use the $STGUT1 utility (MX command) to monitor the number of interrupts buffered by the task supervisor.

XPSSTK= The size of the cross partition stack in 6-byte entries. This stack contains the return address and partition numbers that the supervisor refers to when branching between partition one and supervisor portions in other partitions. You must code a value between 10 and 128. The default size is 20. This default is usually sufficient for your programming needs.

Note: Use the $STGUT1 utility (MX command) to monitor the number of entries in the cross partition stack.

COMMON= A partition 1 supervisor address or the size of the supervisor data area. Code up to eight labels or numbers. The labels represent a partition 1 (address space 0) supervisor address. The numbers represent the size (in 2K blocks) of a partition 1 (address space 0) supervisor data area. The valid numbers are 0 to 31. The syntax of the COMMON= operand is as follows:

```
COMMON=(__,__,__,__,__,__,__,__)
```

Whether you specify labels or numbers, you can map the area as common in all partitions, in any specific partition, or in no partitions. The syntax of the operand is such that each entry (a maximum of eight) corresponds to address spaces 0 through 7. If you do not want to map the common area into a specific partition, code the COMMON= operand as in the following example where the supervisor data areas are mapped into partitions 2, 3, and 5.

SYSTEM - Define Processor Storage *(continued)*

```
COMMON=(EDXSVCX,1,1,0,1)
```

The size of the common area will be rounded upward automatically to a 2K byte boundary.

To map the entire resident supervisor, specify COMMON=EDXSTART. To map only the supervisor data areas, specify COMMON=EDXSVCX. The default, COMMON=EDXSYS, implies no mapping.

INITPRT= The partition into which $INITIAL is loaded after system initialization. If you specify a partition and there is not enough storage in that partition to contain $INITIAL, the system cannot load it and issues a LOAD instruction return code of 70. If you do not specify a partition number and allow INITPRT to default, the system attempts to load $INITIAL into the first partition it finds with enough storage to contain $INITIAL.

INITMOD= A list of entry point names in modules to be executed during system initialization. The modules must be link edited with the supervisor and may not execute LOAD instructions. Each module must return to the entry point INITEXIT. Each module must begin execution in Event Driven Language. Control is passed to the module using a GOTO instruction.

MECBLST= The maximum number (in decimal) of event control blocks (ECBs) allowed in the system at any given time. The MECB statement is used in conjunction with the WAITM instruction to permit a program to wait on the occurrence of multiple events. You can specify from 1 to 64 ECBs. The system generates an ECB pointer consisting of two words (address and address key). For example, if you specify 64, the system generates 128 words.

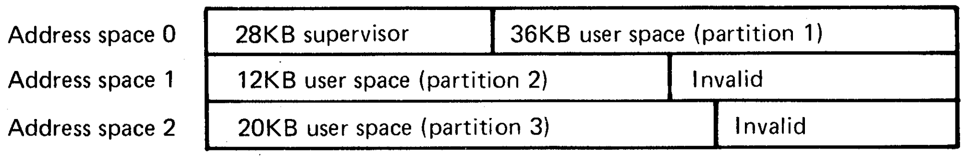
SYSTEM - Define Processor Storage *(continued)*

Example 1: Define a three-partition system on a 96K-byte 4955.

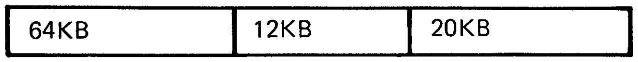
```
SYSTEM    MAXPROG=(3,2,3),PARTS=(32,6,10)
```

The system logically and physically maps in storage as follows:

Logical mapping:


Address space 0
28KB supervisor
36KB user space (partition 1)
Address space 1
12KB user space (partition 2)
Invalid
Address space 2
20KB user space (partition 3)
Invalid

Physical mapping:



1. Partition 1 requires 14 blocks (28KB) for the supervisor and has 18 blocks (36KB) left for user space. Partition 1 can execute up to three programs concurrently.

   Partition 1 is defined as 9 blocks (18KB) of storage. The supervisor required 14 blocks (28KB) and the initialization routines required 9 blocks (18KB) of storage at IPL time. After initialization, the 9 blocks of storage required by the initialization routines were given back as user space.

2. Partition 2 requires 6 blocks (12KB) of storage for user space. Partition 2 can execute up to two programs concurrently.

3. Partition 3 requires 10 blocks (20KB) of storage for user space. Partition 3 can execute up to three program concurrently.

4. There is no unmapped storage.

Note: The 28KB supervisor size is used for illustrative purposes only.

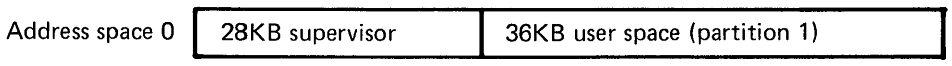
SYSTEM - Define Processor Storage *(continued)*

Example 2: Define a single-partition (64K) system.

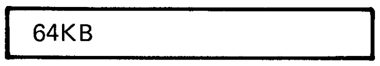
```
         SYSTEM   MAXPROG=5
```

The system logically and physically maps in storage as follows:

Logical mapping:



Physical mapping:



The supervisor requires 14 blocks (28KB) for the supervisor and has 18 blocks (36KB) left for user space. Up to five programs can execute concurrently.

Note: The 28KB supervisor size is used for illustrative purposes only.

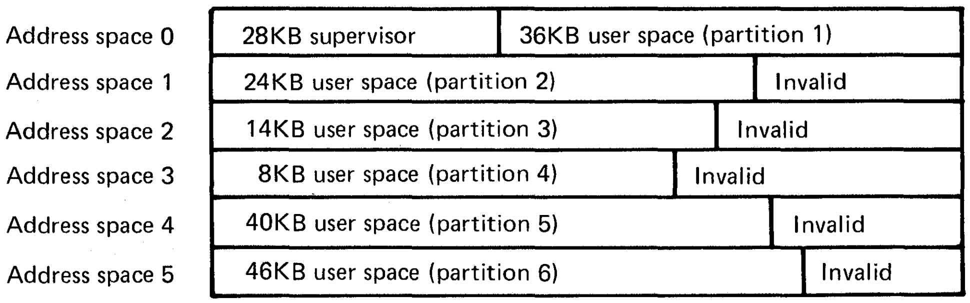
SYSTEM - Define Processor Storage *(continued)*

Example 3: Define a six-partition system on a 196K-byte 4955.

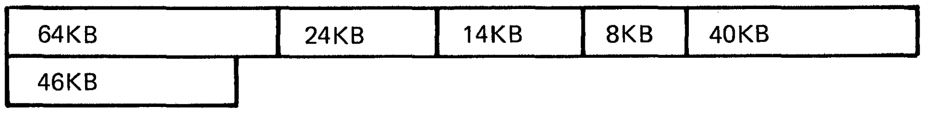
```
SYSTEM    MAXPROG=(1,2,1,3,4,1),                    C
          PARTS=(9,12,7,4,20,23),INITPRT=2
```

The system logically and physically maps in storage as follows:

Logical mapping:


Address space 0
28KB supervisor
36KB user space (partition 1)
Address space 1
24KB user space (partition 2)
Invalid
Address space 2
14KB user space (partition 3)
Invalid
Address space 3
8KB user space (partition 4)
Invalid
Address space 4
40KB user space (partition 5)
Invalid
Address space 5
46KB user space (partition 6)
Invalid

Physical mapping:



SYSTEM - Define Processor Storage *(continued)*

1. Partition 1 requires 14 blocks (28KB) for the supervisor and has 18 blocks (36KB) left for user space. Partition 1 can execute one program at a time.

   Partition 1 is defined as 9 blocks (18KB) of storage. The supervisor required 14 blocks (28KB) and the initialization routines required 9 blocks (18KB) of storage at IPL time. After initialization, the 9 blocks of storage required by the initialization routines were given back as user space.

2. Partition 2 requires 12 blocks (24KB) of storage and can execute up to two program concurrently.

3. Partition 3 requires 7 blocks (14KB) of storage and can execute one program at a time.

4. Partition 4 requires 4 blocks (8KB) of storage and can execute up to three programs concurrently.

5. Partition 5 requires 20 blocks (40KB) of storage and can execute up to four programs concurrently.

6. Partition 6 requires 23 blocks (46KB) of storage and can execute one program at a time.

7. $INITIAL, if it exists, will be loaded into partition 2 if there is enough space.

Note: The 28KB supervisor size is used for illustrative purposes only.

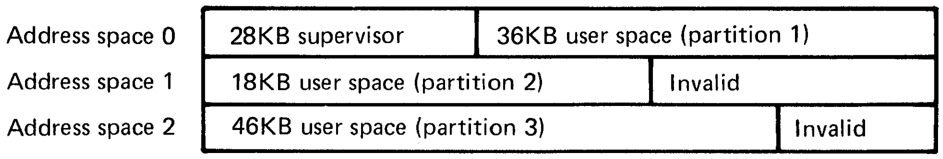
SYSTEM - Define Processor Storage *(continued)*

Example 4: Define a three-partition system on a 128K-byte 4955.

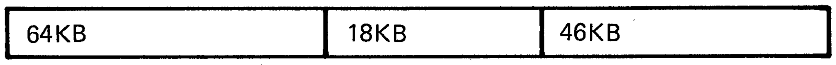
```
         SYSTEM    MAXPROG=(10,10,10),                    C
               PARTS=(27,9,23),INITMOD=(CSXINIT)
```

The system logically and physically maps as follows:

Logical mapping:


Address space 0
28KB supervisor
36KB user space (partition 1)
Address space 1
18KB user space (partition 2)
Invalid
Address space 2
46KB user space (partition 3)
Invalid

Physical mapping:



1. Partition 1 requires 14 blocks (28KB) for the supervisor and has 18 blocks (36KB) left for user space. Partition 1 can execute one program at a time.

   Partition 1 is defined as 27 blocks (54KB) of storage. The supervisor required 14 blocks (28KB) and the initialization routines required 9 blocks (18KB) of storage at IPL time. After initialization, the 9 blocks of storage required by the initialization routines were given back as user space. Although partition 1 was defined as 54KB of user space, the supervisor required 28KB of storage leaving only 36KB of available user space.

2. Partition 2 requires 9 blocks (18KB) of storage and can execute up to 10 programs concurrently.

3. Partition 3 requires 23 blocks (46KB) of storage and can execute up to 10 programs concurrently.

4. The module with entry point name of CSXINIT will be activated during initialization.

Note: The 28KB supervisor size is used for illustrative purposes only.

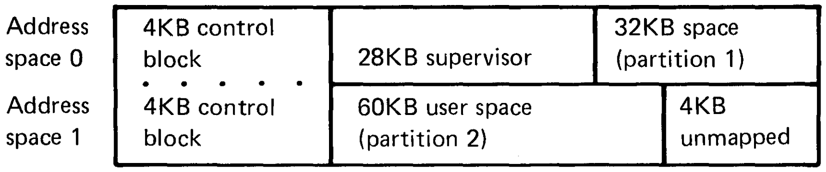
SYSTEM - Define Processor Storage *(continued)*

Example 5: Define a two-partition system on a 128K-byte 4952.

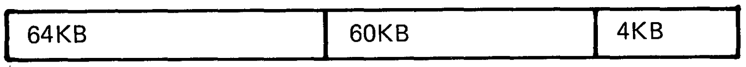
```
SYSTEM    MAXPROG=(3,6),PARTS=(32,32),               C
          COMMON=EDXSVCX,DATEFMT=MMDDYY
```

The system logically and physically maps as follows:

Logical mapping:



Physical mapping:



1. Partition 1 is 32KB and can execute up to three programs concurrently.
2. Partition 2 is 60KB and can execute up to six programs concurrently. The programs all have direct addressability to supervisor control blocks (for example, the CVT (communications vector table) and DVT (device vector table) because of the COMMON=EDXSVCX parameter.
3. Total storage used: 4KB + 28KB + 32KB + 60KB = 124KB. The size of process storage is 128KB leaving 4KB of unmapped storage.

Note: The 28KB supervisor size and the 4KB control block size are used for illustrative purposes only.

SYSTEM - Define Processor Storage *(continued)*

Example 6: Define a two-partition system on a 128K-byte 4952.

```
SYSTEM   MAXPROG=(4,4),PARTS=(32,32),           C
         DATEFMT=MMDDYY
```

The system logically and physically maps as follows:

Logical mapping:

Address space 0	30KB supervisor	34KB user space (partition 1)
Address space 1	64KB user space (partition 2)	

Physical mapping:

64KB	64KB

1. Partition 1 requires 15 blocks (30KB) of storage for the supervisor and has 17 blocks (34KB) of storage left for user space. Partition 1 can execute up to four programs concurrently.

2. Partition 2 requires 32 blocks (64KB) of storage and can execute up to four programs concurrently.

3. When the date is displayed, it will be in month, day, and year format.

Note: The 30KB supervisor size is used for illustrative purposes only.

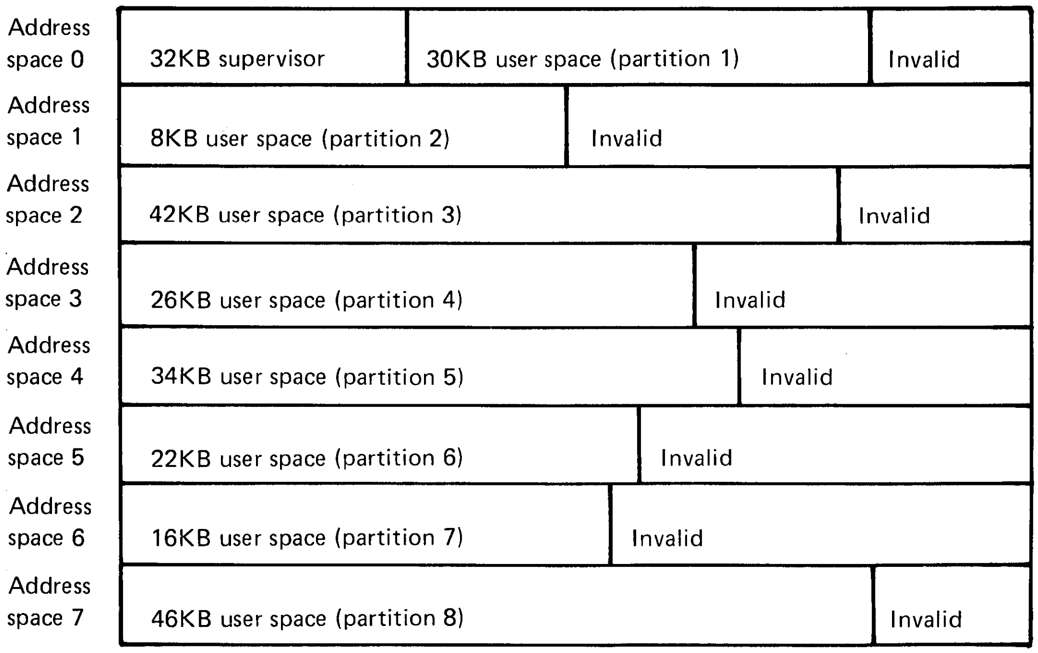
SYSTEM - Define Processor Storage *(continued)*

Example 7: Define an eight-partition system on a 256K-byte 4955.

```
SYSTEM   MAXPROG=(3,1,5,2,2,1,1,4),          C
      PARTS=(15,4,21,13,17,11,8,23)
```

The system logically and physically maps as follows:

Logical mapping:


Address space 0
32KB supervisor
30KB user space (partition 1)
Invalid
Address space 1
8KB user space (partition 2)
Invalid
Address space 2
42KB user space (partition 3)
Invalid
Address space 3
26KB user space (partition 4)
Invalid
Address space 4
34KB user space (partition 5)
Invalid
Address space 5
22KB user space (partition 6)
Invalid
Address space 6
16KB user space (partition 7)
Invalid
Address space 7
46KB user space (partition 8)
Invalid

SYSTEM - Define Processor Storage *(continued)*

Physical mapping:

62KB	8KB	42KB	26KB
34KB	22KB	16KB	46KB

1. Partition 1 requires 16 blocks (32KB) of storage for the supervisor and has 15 blocks (30KB) of storage left for user space. Partition 1 can execute up to three programs concurrently.

2. Partition 2 requires 4 blocks (8KB) of storage for user space and can execute one program at a time.

3. Partition 3 requires 21 blocks (42KB) of storage for user space and can execute up to five programs concurrently.

4. Partition 4 requires 13 blocks (26KB) of storage for user space and can execute up to two programs concurrently.

5. Partition 5 requires 17 blocks (34KB) of storage for user space and can execute up to two programs concurrently.

6. Partition 6 requires 11 blocks (22KB) of storage for user space and can execute one program at a time.

7. Partition 7 requires 8 blocks (16KB) of storage for user space and can execute one program at a time.

8. Partition 8 requires 23 blocks (46KB) of storage for user space and can execute up to four programs concurrently.

Note: The 32KB supervisor size is used for illustrative purposes only.

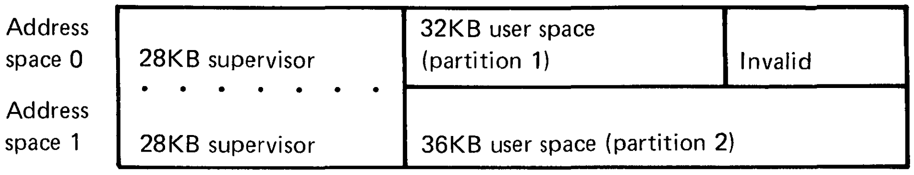
SYSTEM - Define Processor Storage *(continued)*

Example 8: Define a two-partition system on a 96K-byte 4952.

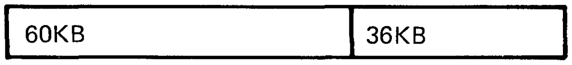
```
SYSTEM    MAXPROG=(3,4),PARTS=(16,18),                    C
          COMMON=EDXSTART
```

The system logically and physically maps as follows:

Logical mapping:



Physical mapping:



1. Because COMMON=EDXSTART was specified, the supervisor is mapped in both partition 1 and partition 2, providing direct addressability to the supervisor for all programs that execute on this system.

2. Partition 1 requires 14 blocks (28KB) of storage for the supervisor and has 16 blocks (32KB) of storage left for user space. Partition 1 can execute up to three programs concurrently.

3. Partition 2 requires 14 blocks (28KB) of storage for the supervisor and has 18 blocks (36KB) of storage left for user space. Partition 2 can execute up to four programs concurrently.

4. Total storage used: 28KB + 32KB + 36KB = 96KB

Note: The 28KB supervisor size is used for illustrative purposes only.

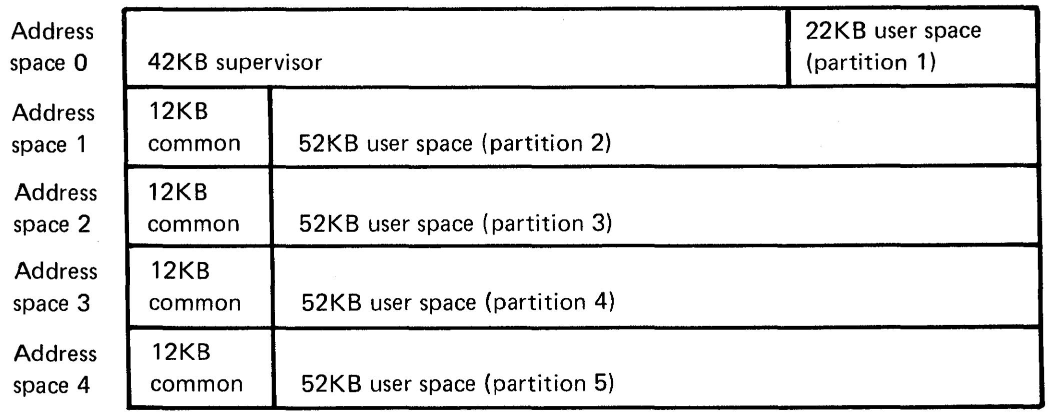
SYSTEM - Define Processor Storage *(continued)*

Example 9: Define a five-partition system on a 256-byte 4955.

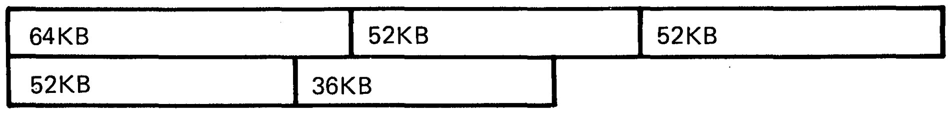
```
SYSTEM   MAXPROG=(10,10,10,10,10),                        C
      PARTS=(26,26,26,26,26),COMMON=USRMAP
```

The supervisor requires 42K bytes of storage and the COMMON area requires 12K bytes of storage. The system logically and physically maps as follows:

Logical mapping:


Address space 0
42KB supervisor
22KB user space (partition 1)
Address space 1
12KB common
52KB user space (partition 2)
Address space 2
12KB common
52KB user space (partition 3)
Address space 3
12KB common
52KB user space (partition 4)
Address space 4
12KB common
52KB user space (partition 5)

Physical mapping:



1. Partition 1 requires 21 blocks (42KB) for the supervisor and has 11 blocks (22KB) left for user space. Partition 1 can execute up to 10 programs concurrently.

2. Partition 2 requires 6 blocks (12KB) to serve as addressing pointers to the COMMON area and has 26 blocks (52KB of actual storage mapped for user space. The 6 blocks of pointers cannot map actual storage since they are reserved as COMMON area pointers. Partition 2 can execute up to 10 programs concurrently.

3. Partitions 3 and 4 are the same as Partition 2.

4. Partition 5 maps all of the actual storage not previously mapped. The six 2KB blocks of storage from partitions 2, 3, and 4 equal 36KB mapped into the last partition. Partition 5 can execute up to 10 programs concurrently.

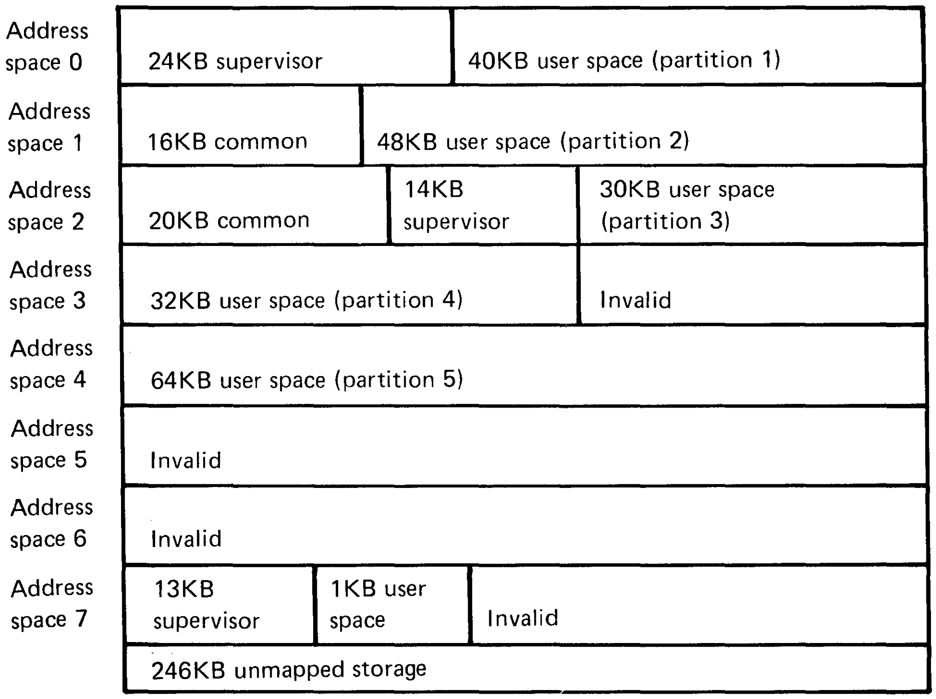
SYSTEM - Define Processor Storage *(continued)*

Example 10: Define a five-partition system on a 512-byte 4956. A multipartition supervisor was generated for this system with 24K of supervisor code in partition 1, 14K in partition 3, and 13K in partition 8.

```
         SYSTEM   MAXPROG=(1,5,5,5,5),PARTS=(32,32,32,16,32)    C
               COMMON=(0,8,10)
```

The system logically and physically maps as follows:

Logical mapping:



SYSTEM - Define Processor Storage *(continued)*

Physical mapping:

64KB	48KB	44KB	32KB
64KB	14KB	246KB unmapped	

1. Partition 1 requires 12 blocks (24KB) for the supervisor and has 20 blocks (40KB) left for user space. Partition 1 can execute up to 10 programs concurrently.

2. Partition 2 requires 8 blocks (16KB) to serve as addressing pointers to the COMMON area and has 24 blocks (48KB) of actual storage mapped for user space. The 8 blocks of pointers cannot map actual storage since they are reserved as COMMON area pointers. Partition 2 can execute up to 10 programs concurrently.

3. Partition 3 requires 10 blocks (20KB) to serve as addressing pointers to the COMMON area and has 7 blocks (14KB) for the supervisor and 15 blocks (30KB) left for user space even though the user requested 64KB of user space. Partition 3 can execute up to 10 programs concurrently.

4. Partition 4 requires 16 blocks (32KB) of storage for user space.

5. Partition 5 is not defined.

6. Partition 6 is not defined.

7. Partition 7 was not defined on the SYSTEM statement, however, a multipartition supervisor was defined with the PART= statement in the $LNKCNTL data set as having 13KB of supervisor code in partition 8. As a result, partition 8 was automatically defined and 13KB of storage was assigned to the supervisor. In addition, because storage is assigned in 2K blocks, 1KB of storage is assigned as user space for a total of 14KB in partition 8.

8. The size of process storage is 512KB. Only 266KB of storage was defined leaving 246KB of unmapped storage.

SYSTEM - Define Processor Storage *(continued)*

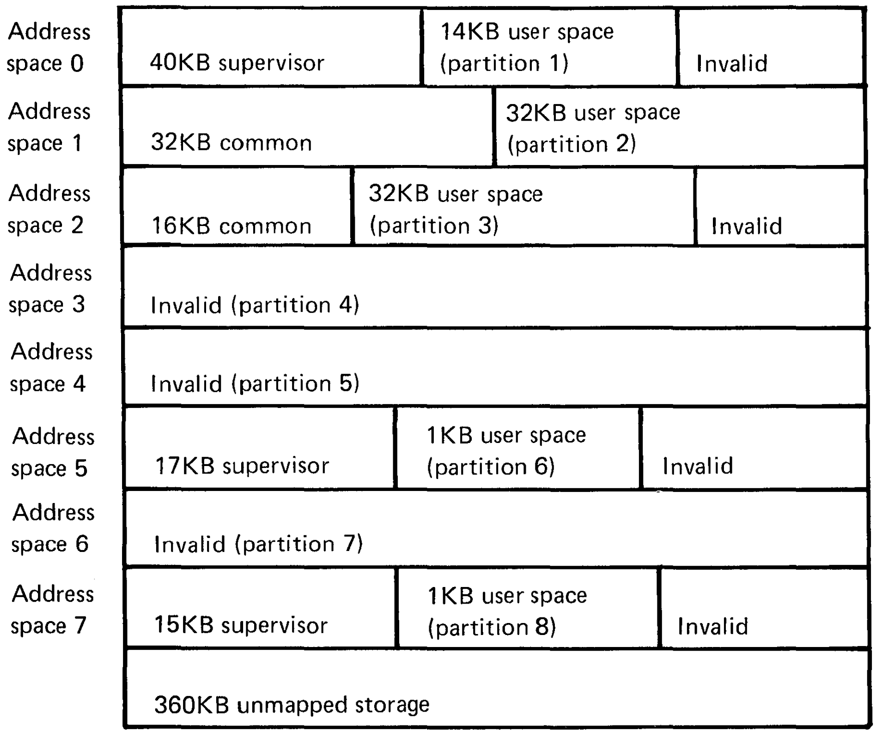
Example 11: Define a three-partition system on a 512-byte 4956. A multipartition supervisor was generated for this system with 50K of supervisor code in partition 1, 17K in partition 6, and 15K in partition 8.

```
SYSTEM    MAXPROG=(1,10,1),PARTS=(2,16,16),         C
          COMMON=(0,LABEL,8)
```

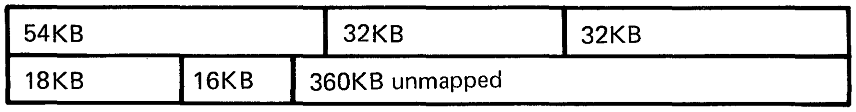
In this example, LABEL is at address X'8000' of the supervisor in partition 1.

The system logically and physically maps as follows:

Logical mapping:



Physical mapping:



SYSTEM - Define Processor Storage *(continued)*

1. Partition 1 requires 20 blocks (40KB) for the supervisor and has 7 blocks (14KB) left for user space. Partition 1 can execute one program. After initialization, the 5 blocks (10KB) of storage required by the initialization routines were given back as user space.

2. Partition 2 requires 16 blocks (32KB) to serve as addressing pointers to the COMMON area and has 16 blocks (32KB) of actual storage mapped for user space. The 16 blocks of pointers cannot map actual storage since they are reserved as COMMON area pointers. Partition 2 can execute up to 10 programs concurrently.

3. Partition 3 requires 8 blocks (16KB) to serve as addressing pointers to the COMMON area and has 16 blocks (32KB) for the supervisor and 15 blocks (30KB) for user space. Partition 3 can execute up to 10 programs concurrently.

4. Partition 4 is not defined.

5. Partition 5 is not defined.

6. Partition 6 was not defined on the SYSTEM statement, however, a multipartition supervisor was defined with the PART= statement in the $LNKCNTL data set as having 17KB of supervisor code in partition 6. As a result, partition 6 was automatically defined and 17KB of storage was assigned to the supervisor. In addition, because storage is assigned in 2K blocks, 1KB of storage is assigned as user space for a total of 18KB in partition 6.

7. Partition 7 is not defined.

8. Partition 8 was not defined on the SYSTEM statement, however, a multipartition supervisor was defined with the PART= statement in the $LNKCNTL data set as having 15KB of supervisor code in partition 8. As a result, partition 8 was automatically defined and 15KB of storage was assigned to the supervisor. In addition, because storage is assigned in 2K blocks, 1KB of storage is assigned as user space for a total of 16KB in partition 8.

9. The size of process storage is 512KB. Only 152KB of storage was defined leaving 360KB of unmapped storage.

TAPE - Define Tape Device

TAPE defines tape devices. One TAPE statement is required for each tape device on the system. Group all DISK and TAPE statements together. The last TAPE or DISK statement must specify END=YES.

Syntax:

```
blank        TAPE DEVICE=,ADDRESS=,DENSITY=,LABEL=,ID=,
                  TASK=,END=

Required:    DEVICE=,ADDRESS=,ID=
Defaults:    DENSITY=1600,LABEL=SL,TASK=NO,END=NO
```

Operand	*Description*
DEVICE=	Required if 4968; otherwise, defaults to 4969. **4968** 4968 tape unit **4969** 4969 tape unit
ADDRESS=	A two-digit hexadecimal number specifying the address assigned to the unit.
DENSITY=	Tape density to be used for this device. For the 4968 tape unit, the valid densities are 1600, 3200, and DUAL. When DUAL is coded, density defaults to 1600 BPI. For the 4969 tape unit, the valid densities are 800, 1600, and DUAL. When DUAL is coded, density defaults to 1600 BPI. **Note:** A 4968 density selection of 3200 BPI is not ANSI standard and is only compatible with other 4968 tape units.
LABEL=	Type of processing to be done on this device. Standard label (SL), nonlabel (NL), and bypass label processing (BLP) are the only types supported. The default is LABEL=SL.

TAPE - Define Tape Device *(continued)*

ID= A one- to six-character name that is associated with the device. This operand is used primarily for specifying the drive when NL or BLP is used.

Note: Tape IDS, volume names (see the DISK definition statement), and the labels associated with TERMINAL definition statements must be unique. No volume name or TERMINAL definition statement label can match a tape ID.

TASK= YES, causes a new I/O task to be generated. This task is used to service I/O request for this and subsequent tapes until a new TAPE statement with TASK=YES is encountered. For best performance, specify TASK=YES for each tape unit that has a controller. The default is TASK=NO; one task is received.

Additional storage required for each TASK=YES is 128 bytes.

END= YES, for the last statement in the DISK/TAPE sequence. The default is END=NO.

Example: Define a 4969 tape unit at address 4C. The tape is a standard label tape set at a density of 1600. The name associated with the tape unit is $TAPE1.

```
         TAPE  DEVICE=4969,ADDRESS=4C,DENSITY=1600,LABEL=SL,      C
               ID=$TAPE1,TASK=YES,END=YES
```

Note: END=YES is required only once for the DISK and TAPE definition statements.

TERMINAL - Define Input/Output Terminals

TERMINAL defines each EDX terminal to be supported in the generated system. It is coded in your source statements for system generation, and is assembled with DISK, SYSTEM, and other supervisor definition statements. The DEVICE= operand of the TERMINAL statement identifies the type of terminal.

EDX terminals include a wide variety of devices such as keyboard/displays, printers, other processors, plotters and lab equipment. The operands of the TERMINAL statement define many of the characteristics of a particular terminal. The first operand usually coded is DEVICE= and its value determines which operands to code and their values. One such operand whose inclusion and value depends upon the value of the DEVICE operand is the CODTYPE operand. The devices supported and the feature code used to connect the device to the Series/1 are shown in Figure 12 .

Terminal	Feature number	DEVICE operand of TERMINAL	CODTYPE operand
IBM 3101 Display Terminal, all models	#1310, #2095 with #2096, 2095 with RPQ D02350	ACCA	ASCII
IBM 3101 Display Terminal, all models	#1610, #2091 with #2092	ACCA	EBASC
IBM 3101 Display Terminal, models 10, 12, and 13	#7850	TTY[3]	ASCII
IBM 4978 Display Station	RPQ D02038	4978	N/A
IBM 4979 Display Station	#3585	4979	N/A
IBM 4980 Display Station	#1250	4980	N/A
Teletype[3] ASR 33/35 (TTY) or equivalent	#7850	TTY	ASCII
ASCII Terminal	#2095 with #2096	ACCA	ASCII
ASCII Terminal	#1610, #2091 with #2092	ACCA	EBASC
IBM 2741 Communication Terminal	#1610	2741	CRSP or EBCD
IBM 4973 Line Printer	#5630	4973	N/A

Figure 12 (Part 1 of 2). Supported EDX terminals and DEVICE/CODTYPE operands

[3] Trademark of Teletype Corporation

TERMINAL - Define Input/Output Terminals *(continued)*

Terminal	Feature number	DEVICE operand of TERMINAL	CODTYPE operand
IBM 4974 Matrix Printer	#5620	4974	N/A
IBM 4975 Printer (all models except 01A)	#1310	4975-01R 4975-01L 4975-02R 4975-02L	N/A
IBM 4975-01A Printer	#1310 or #2095 with 2096	ACCA	ASCII
IBM 5219 Line Printer	#5640	5219	N/A
IBM 5224 Matrix Printer	#5640	5224	N/A
IBM 5225 Matrix Printer	#5640	5225	N/A
IBM Series/1	#1610, RPQs D02241 and D02242	PROC	N/A
General Purpose Interface Bus (GPIB)	RPQ D02118	GPIB	ASCII
Textronics 40xx Graphics Terminal	#1560	4013	ASCII

Figure 12 (Part 2 of 2). Supported EDX terminals and DEVICE/CODTYPE operands

ASCII Transmission Codes

Terminals and other devices equivalent to the Teletype ASR 33/35 are referred to as ASCII terminals. Depending upon the feature number used to connect these terminals to the Series/1, the transmission code is either ASCII or EBASC (mirror image ASCII). The CODTYPE operand specifies the transmission code to be used by the terminal and defines the internal representation of characters. (See *Communications Guide* for a list of transmission codes.)

The ASCII transmission code, in which the characters appear in main storage in ASCII code, is used by features #1310, #7850, #2095, and #2096.

TERMINAL - Define Input/Output Terminals *(continued)*

The other transmission code is EBASC (mirror image ASCII) and is used by the 1610 and 2091 controllers and the 2092 adapter. EBASC is the mirror image within a ***byte*** of the ASCII representation; the bits appear swapped end-for-end within each byte. For example, the character "1" would look as follows (internally):

	Hexadecimal	Binary
ASCII	X'31'	0 0 1 1 0 0 0 1
EBASC (mirror image ASCII)	X'8C'	1 0 0 0 1 1 0 0

Note that the bit representation of a character appearing at the terminal is the same for all attachments and the bit representation for all characters in the Series/1 is ASCII. However, depending upon the attachment, ASCII or EBASC is used as the transmission code.

Note also that EDX supports "no parity" only, but some ASCII terminals use even or odd. For information on the parity of a particular terminal, refer to the associated hardware manual. If you require even or odd parity, use the EXIO facility.

A terminal can be directly attached (RS-422 or LOCAL) to the Series/1 or attached through a modem (remote). A remote terminal is connected to the Series/1 by an asynchronous (start/stop) communications line. The line can be point-to-point switched (SWITCHED) or point-to-point nonswitched (PTTOPT). The TERMCTRL instruction has operands that are used to control the modem; see the *Language Reference.*

Before preparing TERMINAL statements, you need to know the characteristics of your terminals, the way they will be attached to your Series/1, and how you plan to use them in your application. (Review the appropriate hardware manuals, and refer to the *Language Reference* and the *Communications Guide.*)

Symbolic Reference to Terminals

The label on the TERMINAL statement assigns a name to the device for purposes of reference by the application program. Four such names have special meaning to the supervisor and should be assigned to the appropriate device:

$SYSLOG	Names the system logging device or operator station, and must be defined in every system. The starter supervisor defines $SYSLOG as a 4978 display station. $SYSLOG is the primary device to receive/display all exception messages.
$SYSLOGA	Names the alternate system logging device. If unrecoverable errors prevent use of $SYSLOG, the system will use the $SYSLOGA terminal as the system logging device/operator station. If defined, this device should be a terminal with keyboard capability, not just a printer. The starter supervisor defines the $SYSLOGA terminal as a teletypewriter device.

TERMINAL - Define Input/Output Terminals *(continued)*

$SYSLOGB Names the second alternate system logging device. If unrecoverable errors prevent use of $SYSLOG, the system will use the $SYSLOGA terminal as the system logging device/operator station. If no $SYSLOGA terminal is defined or unrecoverable errors prevent the use of $SYSLOGA, the system will use the $SYSLOGB terminal as the system logging device/operator station. If defined, this device should be a terminal with keyboard capability, not just a printer. The starter supervisor defines the $SYSLOGB terminal as a 3101 Display Terminal in block mode.

$SYSPRTR Names the system printer. If defined, the hard copy output from all system programs are directed to this device. The starter supervisor defines a 4974 matrix printer as the $SYSPRTR device.

Assignment of a name to a terminal designates that terminal as a global resource to be accessed by any application program through use of the ENQT and IOCB instructions described in the *Language Reference*.

Coding the TERMINAL Statement

In the following matrix, the DEVICE operand is listed across the top of the matrix. The operands that can be specified for a particular device as determined by the DEVICE operand are indicated by an **X**.

Following the matrix, the syntax and operands for each supported device are described by device type.

Notes:

1. All TERMINAL statements must be grouped together with the last statement including an END=YES specification.

2. Specify TYPE=DSECT on the TERMINAL statement to include the EDL/copy code when assembling under $S1ASM and using the Macro Library.

3. Labels on TERMINAL statements, tape IDs (see the TAPE definition statement), and volume names (see the DISK definition statement must be unique. Volume names or tape IDs must not match the label on a TERMINAL statement.

4. The maximum LINSIZE is device dependent but can never by greater than 254.

If you use the Remote Management Utility and need the PASSTHRU function, two virtual terminals are required. (See "Examples and Defaults" on page IS-237 for a sample configuration.) (For a detailed description of the PASSTHRU function, see the Remote Management Utility chapter in *Communications Guide.*)

TERMINAL - Define Input/Output Terminals *(continued)*

	Device operand of TERMINAL statement																
Parameter	2741	4013	4973	4974	4975	5219	5225	5224	4978	4979	4980	ACCA	TTY	PROC	VIRT	GPIB	S1S1
ADAPTER	X				X	X	X	X			X	X					
ADDRESS	X		X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
ATTN									X	X	X	X	X			X	
BITRATE	X				X						X	X		X			
BOTM	X	X	X	X	X	X	X	X	X	X	X	X	X				
CHARDEL	X	X										X	X				
CHARSET					X	X	X	X									
CHKSUM																	X
COD												X					
CODTYPE	X	X										X	X	X		X	
CR	X	X										X	X	X			
CRDELAY	X	X										X	X	X			
DI/DO/PI		X															
ECHO	X	X											X				
END	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
HDCOPY									X	X	X						
LEFTM	X	X	X	X	X	X	X	X	X	X	X	X	X				
LF	X	X										X	X	X			
LINEDEL	X	X										X	X				
LINSIZE	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
LMODE					X							X					
MODE												X	X				
NHIST									X	X	X	X					
OVFLINE	X	X	X	X	X	X	X	X	X	X	X	X	X				
PAGSIZE	X	X	X	X	X	X	X	X				X	X				
PART	X	X							X	X	X	X	X			X	
PF1									X	X	X	X	X				
PORT						X	X	X			X						
RANGE												X		X			
RIGHTM	X	X	X	X	X	X	X	X	X	X	X	X	X				
SCREEN	X	X							X	X	X	X	X			X	
SECADDR						X	X	X			X						
SPOOL	X		X	X	X	X	X	X	X	X	X	X	X			X	
SYNC															X		
TIMERS												X					
TOPM	X	X	X	X	X	X	X	X	X	X	X	X	X				
TYPE	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)	(2)

Figure 13. Device-Dependent Operands

TERMINAL - 2741

2741 Terminal

A TERMINAL definition statement with DEVICE=2741 defines a 2741 communications terminal attached via 1610 controller.

Syntax:

```
label      TERMINAL    DEVICE=,ADDRESS=,PAGSIZE=,LINSIZE=,CODTYPE=,
                       TOPM=,BOTM=,LEFTM=,RIGHTM=,OVFLINE=,
                       LINEDEL=,CHARDEL=,CRDELAY=,ECHO=,BITRATE=,
                       ADAPTER=,CR=,LF=,SCREEN=,PART=,SPOOL=,END=

 Required:  DEVICE=,ADDRESS=
 Defaults:  PAGSIZE=66,LINSIZE=130,CODTYPE=EBCD,TOPM=0,BOTM=65,
            LEFTM=0,RIGHTM=129,SCREEN=NO,OVFLINE=NO,LINEDEL=AO,
            CHARDEL=5D,CRDELAY=0,ECHO=YES,CR=5B,LF=5B,BITRATE=134,
            ADAPTER=SINGLE,PART=1,SPOOL=NO,END=NO
```

Operand	***Description***
DEVICE=	2741 communications terminal attached via 1610 controller.
ADDRESS=	The address (in hexadecimal) of the device.
PAGSIZE=	The physical page size (form length) of the I/O medium. Specify a decimal number between 1 and the maximum value which is meaningful for the device. The default is PAGSIZE=66.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate, but the value cannot be increased dynamically. The default is LINSIZE=130.
CODTYPE=	The transmission code used by the terminal. Specify either EBCD (PTTC/EBCD) or CRSP (PTTC/correspondence). The default is CODTYPE=EBCD.
TOPM=	The top margin (a decimal number between zero and PAGSIZE-1) to indicate the top of the logical page within the physical page for the device. The default is TOPM=0.
BOTM=	The bottom margin, the last usable line on a page. Its value must be between TOPM+NHIST and PAGSIZE-1. If an output instruction would cause the line number to increase beyond this value, then a page eject, or wrap to line zero, is performed before the operation is continued. The default is BOTM=65.
LEFTM=	The left margin, the character position at which input or output will begin. Specify a decimal value between zero and LINSIZE-1. The default is LEFTM=0.

2741 Terminal *(continued)*

RIGHTM=	A value (between LEFTM and LINSIZE-1) that determines the last usable character position within a line. Position numbering begins at zero. The default is RIGHTM=129.
OVFLINE=	YES, if output lines which exceed the right margin are to be continued on the next line. The default is OVFLINE=NO. The overflow condition occurs when the system buffer (or a buffer in an application program) becomes full and the application program has taken no action to write the buffer to the device.
LINEDEL=	A two-digit hexadecimal character that defines the character to be entered to restart an input line. In some cases, input of this character causes a repeat of the previous output message. The default is LINEDEL=AO.
CHARDEL=	A two-digit hexadecimal character which indicates deletion of the previous input character. It is meaningful only for devices whose mode of transmission is one character at a time, as described in the LINEDEL operand. The default is CHARDEL=5D. **Note:** When CHARDEL= and LINEDEL= values are equal, no translation and editing will occur.
CRDELAY=	The number of idle times required for a carriage return to complete for teletypewriter devices. If printing occurs during the carriage return, CRDELAY is too small. The default is CRDELAY=0.
ECHO=	NO, for devices that do not require input characters to be written back (echoed) by the processor for printing. YES (the default) is appropriate for most devices connected through the teletypewriter adapter.
BITRATE=	The rate (in bits per second) that this terminal will be operating. The default is 134 bits per second.
ADAPTER=	SINGLE, for the single-line controller (the default). For 2741, only SINGLE is allowed. This operand should match the jumpers on the 1610 controller card. (Refer to *Communications Guide* for hardware considerations.)
CR=	The character the system tests in order to determine if a new line function is to be performed. The default is CR=5B.

2741 Terminal *(continued)*

LF= The character the system send to the terminal when a new line function is to be performed. If the same value is coded for LF= as was coded (or defaulted) for CR= then the CR character which terminates an input operation will not be echoed to the terminal; the terminal is assumed to be an auto line-feed device. The default is LF=5B.

SCREEN= YES or ROLL, for screens that are to be operated similar to a typewriter. When the screen is full, you must press the enter key for output to continue.

NO (the default), for hard-copy devices. When the screen fills up, you do ***not*** have to press enter for output to continue.

PART= The partition (1–8) with which the terminal is normally associated. The default is partition 1.

SPOOL= YES, to define the terminal as a valid spoolable device.

NO (the default), to specify that the terminal is not a valid spoolable device. However, the spoolable characteristics of the terminal can be redefined at a later time using the $TERMUT1 utility.

END= YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

Examples and Defaults

In the following example, the default assignments for DEVICE=2741 are shown as if they were explicitly coded in the TERMINAL statement. If a operand is not shown, then it is not relevant for the device. Address assignment is for illustration only.

Example: IBM 2741 terminal TERMINAL statement.

```
label    TERMINAL  DEVICE=2741,ADDRESS=08
```

is equivalent to:

```
label    TERMINAL  DEVICE=2741,ADDRESS=08,PAGSIZE=66,LINSIZE=130, C
               CODTYPE=EBCD,TOPM=0,BOTM=65,LEFTM=0,RIGHTM=129,    C
               SCREEN=NO,OVFLINE=NO,LINEDEL=AO,CHARDEL=5D,        C
               CRDELAY=0,ECHO=YES,CR=5B,LF=5B,BITRATE=134,        C
               ADAPTER=SINGLE,PART=1,SPOOL=NO,END=NO
```

4013 Terminal

A TERMINAL definition statement with DEVICE=4013 defines a Tektronix 4013 graphics terminal or similar device attached via 1560 adapter.

Terminal support is provided for digital I/O devices such as the Tektronix 4000 Series of Display Terminals equipped with the General Purpose Parallel Interface (Tektronix Custom Feature Number CM021-0109-03) or terminals having equivalent hardware interfaces. (Refer to the *Communications Guide.*)

Syntax:

```
label       TERMINAL     DEVICE=,PAGSIZE=,LINSIZE=,CODTYPE=,
                         TOPM=,BOTM=,LEFTM=,RIGHTM=,OVFLINE=,
                         LINEDEL=,CHARDEL=,CRDELAY=,ECHO=,
                         CR=,LF=,SCREEN=,PART=,DI=,DO=,PI=,
                         END=

Required:   DEVICE= ,and DI=,DO=, or PI=
Defaults:   PAGSIZE=35,LINSIZE=72,CODTYPE=ASCII,TOPM=0,BOTM=34,
            LEFTM=0,RIGHTM=71,SCREEN=NO,OVFLINE=NO,LINEDEL=7F,
            CHARDEL=08,CRDELAY=0,ECHO=YES,CR=0D,LF=0A,PART=1,END=NO
```

Operand	***Description***
DEVICE=	4013 (or other 4000 Series) graphics terminal attached via 1560 adapter.
PAGSIZE=	The physical page size (form length) of the I/O medium. Specify a decimal number between 1 and the maximum value which is meaningful for the device. For screen devices, specify the size of the screen in lines. The default is PAGSIZE=35.
CODTYPE=	The transmission code used by the terminal; in this case, ASCII.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate (for example, 80 for the 4013 graphics terminal), but the value cannot be increased dynamically. The default is LINSIZE=72.
TOPM=	The top margin (a decimal number between zero and PAGSIZE-1) to indicate the top of the logical page within the physical page for the device. The default is TOPM=0.
BOTM=	The bottom margin, the last usable line on a page. Its value must be between TOPM+NHIST and PAGSIZE-1. If an output instruction would cause the line number to increase beyond this value, then a page eject, or wrap to line zero, is performed before the operation is continued. The default is BOTM=34.

4013 Terminal *(continued)*

LEFTM=	The left margin, the character position at which input or output will begin. Specify a decimal value between zero and LINSIZE-1. The default is LEFTM=0.
RIGHTM=	A value (between LEFTM and LINSIZE-1) that determines the last usable character position within a line. Position numbering begins at zero. The default is RIGHTM=71.
OVFLINE=	YES, if output lines which exceed the right margin are to be continued on the next line. The default is OVFLINE=NO. The overflow condition occurs when the system buffer (or a buffer in an application program) becomes full and the application program has taken no action to write the buffer to the device.
LINEDEL=	A two-digit hexadecimal character that defines the character to be entered to restart an input line. In some cases, input of this character causes a repeat of the previous output message. The default is LINEDEL=7F. For information on how to code this operand for ASCII terminals, refer to "ASCII Transmission Codes" on page IS-184.
CHARDEL=	A two-digit hexadecimal character which indicates deletion of the previous input character. It is meaningful only for devices whose mode of transmission is one character at a time, as described in the LINEDEL operand. The default is CHARDEL=08. **Note:** When CHARDEL= and LINEDEL= values are equal, no translation or editing will occur.
CRDELAY=	The number of idle times required for a carriage return to complete for teletypewriter devices. If printing occurs during the carriage return, CRDELAY is too small. The default is CRDELAY=0.
ECHO=	NO, for devices that do not require input characters to be written back (echoed) by the processor for printing. YES (the default) is appropriate for most devices connected through the teletypewriter adapter. See the LF parameter description regarding suppression of the echo of the CR character.
CR=	The character to be tested to determine if a new line function is to be performed. The default is CR=0D. For further information on how to code this operand for ASCII terminals, refer to "ASCII Transmission Codes" on page IS-184.

4013 Terminal *(continued)*

LF=	The character to be sent to the terminal when a new line function is to be performed. If the same value is coded for LF= as was coded (or defaulted) for CR= then the CR character which terminates an input operation will not be echoed to the terminal; the terminal is assumed to be an auto line-feed device. The default is LF=0A. For further information on how to code this operand for ASCII terminals, refer to "ASCII Transmission Codes" on page IS-184.
SCREEN=	YES or ROLL, for screens that are to be operated similar to a typewriter. When the screen is full, you must press the enter key for the output to continue. NO (the default), for hard-copy devices. When the screen fills up, you do ***not*** have to press enter for the output to continue.
DI=(address,termaddr)	address - The digital input group address or termaddr - The hardware subaddress (0–7) of the terminal defining the value used to select the terminal for digital input
DO=(address,termaddr)	address - The digital output group address. or termaddr - The hardware subaddress (0-7) to define the digital output subaddress of the terminal.
PI=(address,bit)	address - The process interrupt group address. or bit - The bit (0–15) to define the particular interrupting point assigned to the terminal.
PART=	The partition (1–8) with which the terminal is normally associated. The default is partition 1. You can change the partition assignment at execution time with the $CP operator command described in *Operation Guide*.
END=	YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

4013 Terminal *(continued)*

Example and Defaults

In the following example, the default assignments for DEVICE=4013 are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignment is for illustration only.

Example: Tektronix 4013 (DI/DO parallel interface). TERMINAL Statement

```
label    TERMINAL  DEVICE=4013,DI=(80,01),DO=(87,01),PI=(84,04)
```

is equivalent to:

```
label    TERMINAL  DEVICE=4013,DI=(80,01),DO=(87,01),             C
               PI=(84,04),PAGSIZE=35,LINSIZE=72,                 C
               CODTYPE=ASCII,TOPM=0,BOTM=34,LEFTM=0,             C
               RIGHTM=71,SCREEN=NO,OVFLINE=NO,                   C
               LINEDEL=7F,CHARDEL=08,CRDELAY=0,ECHO=YES,         C
               CR=0D,LF=0A,PART=1,END=NO
```

4973/4974 Printers

A TERMINAL definition statement with DEVICE=4973 (or 4974) defines a 4973 line printer attached via 5630 adapter or a 4974 matrix printer attached via 5620 adapter.

Syntax:

```
label        TERMINAL     DEVICE=,ADDRESS=,PAGSIZE=,LINSIZE=,TOPM=,
                          BOTM=,LEFTM=,RIGHTM=,OVFLINE=,SPOOL=,END=

Required:    DEVICE=,ADDRESS=
Defaults:    PAGSIZE=66,LINSIZE=132,TOPM=3,BOTM=63,LEFTM=0,
             RIGHTM=131,OVFLINE=NO,SPOOL=YES,END=NO
```

Operand	*Description*
DEVICE=	One of the following device types: **4973** 4973 line printer attached via 5630 Adapter **4974** 4974 matrix printer attached via 5620 Adapter
ADDRESS=	The address (in hexadecimal) of the device.
PAGSIZE=	The physical page size (form length) of the I/O medium. Specify a decimal number between 1 and the maximum value which is meaningful for the device. For printers, specify the number of lines per page. The default is PAGSIZE=66.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate (for example, 132 for the 4973 and 4974 printers), but the value cannot be increased dynamically. The default is LINSIZE=132.
TOPM=	The top margin (a decimal number between zero and PAGSIZE-1) to indicate the top of the logical page within the physical page for the device. The default is TOPM=3.
BOTM=	The bottom margin, the last usable line on a page. Its value must be between TOPM+NHIST and PAGSIZE-1. If an output instruction would cause the line number to increase beyond this value, then a page eject, or wrap to line zero, is performed before the operation is continued. The default is BOTM=63.
LEFTM=	The left margin, the character position at which input or output will begin. Specify a decimal value between zero and LINSIZE-1. The default is LEFTM=0.
RIGHTM=	A value (between LEFTM and LINSIZE-1) that determines the last usable character position within a line. Position numbering begins at zero. The default is RIGHTM=131.

4973/4974 Printers *(continued)*

OVFLINE= YES, if output lines which exceed the right margin are to be continued on the next line. The default is OVFLINE=NO.

The overflow condition occurs when the system buffer (or a buffer in an application program) becomes full and the application program has taken no action to write the buffer to the device.

SPOOL= YES (the default), to define terminal as a valid spoolable device.

NO, to specify terminal is not a valid spoolable device. However the spoolable characteristics of the terminal can be redefined at a later time using the $TERMUT1 utility.

END= YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

Example and Defaults

In the following example, the default assignments for DEVICE=4973 (or 4974) are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignment is for illustration only.

Example: 4974 matrix printer TERMINAL statement.

```
label     TERMINAL   DEVICE=4974,ADDRESS=01

is equivalent to:

label     TERMINAL   DEVICE=4974,ADDRESS=01,PAGSIZE=66,          C
                 LINSIZE=132,TOPM=3,BOTM=63,LEFTM=0,              C
                 RIGHTM=131,OVFLINE=NO,SPOOL=YES,END=NO
```

4975 Printer

A TERMINAL definition statement with DEVICE=4975 defines a 4975 matrix printer (connected locally or remotely) via a Mutlifunction Attachment Feature #1310 (MFA).

Note: For the 4975-01A printer, see the section headed TERMINAL - ACCA.

Syntax:

```
label       TERMINAL     DEVICE=,ADDRESS=,PAGSIZE=,LINSIZE=,CHARSET=,
                         TOPM=,BOTM=,LEFTM=,RIGHTM=,OVFLINE=,
                         BITRATE=,LMODE=,ADAPTER=,SPOOL=,END=

Required:   DEVICE=,ADDRESS=
Defaults:   PAGSIZE=66,LINSIZE=132,CHARSET=USCA,TOPM=3,BOTM=63,
            LEFTM=0,RIGHTM=131,OVFLINE=NO,BITRATE=1200,LMODE=LOCAL,
            ADAPTER=MFA,SPOOL=YES,END=NO
```

label Required if ADAPTER=MFA; otherwise optional.

Operand	***Description***
DEVICE=	One of the following device types:

4975-01L or -02L	4975 matrix printer locally attached via #1310 (MFA) Adapter.
4975-01R or -02R	4975 matrix printer remotely attached via #1310 (MFA) Adapter.

ADDRESS= The address (in hexadecimal) of the device.

PAGSIZE= The physical page size (form length) of the I/O medium. Specify a decimal number between 1 and the maximum value which is meaningful for the device. For printers, specify the number of lines per page. The default is PAGSIZE=66.

LINSIZE= The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate, but the value cannot be increased dynamically. The default is LINSIZE=132.

For the 4975, the maximum number of characters contained on a line depends on the print density you specify with the TERMCTRL instruction or the $TERMUT1 utility. For 4975 maximum line widths, see the PDEN= operand of the TERMCTRL instruction in the *Language Reference*.

4975 Printer *(continued)*

CHARSET= Select the character set for the device. Specify one of the following character sets:

AUGE	Austrian and German
BELG	Belgian
BRZL	Brazilian
DNNR	Danish and Norwegian
FRAN	French (France)
FRCA	French (Canadian)
INTL	International (multinational)
ITAL	Italian
JAEN	Japanese/English
KANA	Japanese katakana
PORT	Portuguese
SPAN	Spanish (Spain)
SPNS	Spanish (countries other than Spain)
SWFI	Swedish and Finnish
UKIN	English (United Kingdom)
USCA	English (United States and Canada).

The default character set is USCA. In addition, the characters per inch and the print mode for the 4975 are automatically set as follows:

Model 1	10 characters per inch
Model 2	10 characters per inch (draft mode).

These values remain in effect until you change them using either the TERMCTRL instruction or the $TERMUT1 utility.

TOPM= The top margin (a decimal number between zero and PAGSIZE-1) to indicate the top of the logical page within the physical page for the device. The default is TOPM=3.

BOTM= The bottom margin, the last usable line on a page. Its value must be between TOPM+NHIST and PAGSIZE-1. If an output instruction would cause the line number to increase beyond this value, then a page eject, or wrap to line zero, is performed before the operation is continued. The default is BOTM=63.

LEFTM= The left margin, the character position at which input or output will begin. Specify a decimal value between zero and LINSIZE-1. The default is LEFTM=0.

RIGHTM= A value (between LEFTM and LINSIZE-1) that determines the last usable character position within a line. Position numbering begins at zero. The default is RIGHTM=131.

4975 Printer *(continued)*

OVFLINE= YES, if output lines which exceed the right margin are to be continued on the next line. The default is OVFLINE=NO.

The overflow condition occurs when the system buffer (or a buffer in an application program) becomes full and the application program has taken no action to write the buffer to the device.

BITRATE= The rate (in bits per second) that this terminal will be operating. The BITRATE parameter is valid only for remotely attached 4975 Models 01R and 02R and is not used for a locally attached 4975 Models 01L or 02L.

1200 The default for printers directly connected (LMODE=LOCAL) or running on asynchronous modems. This is the only bit rate allowed for the 4975 Model 01R.

2400 For the 4975 Model 02R set to 2400 bits per second either directly attached or running on asynchronous modems.

4800 For the 4975 Model 02R set to 4800 bits per second either directly attached or running on asynchronous modems.

EXT For the 4975 Model 02R that is running on a synchronous modem.

Note: 4975 internal clocking bit rates are determined by the switch settings on the printer.

LMODE= For 4975 R (RS-232) support connected via the #1310 attachment, there are two options: LOCAL and PTTOPT. The default is LMODE=LOCAL.

LOCAL means a 4975 R model is directly connected to a Series/1 with no modems. PTTOPT means a 4975 R model connected to a Series/1 on a leased line with internal (1200 BPS) or external (1200, 2400, or 4800 BPS) modems.

This parameter is not valid for 4975 L (RS-422) support.

ADAPTER= MFA, for a Multifunction Attachment Feature #1310. (The 1310 Multifunction Attachment is defined with the ADAPTER definition statement.)

SPOOL= YES (the default), to define terminal as a valid spoolable device.

NO, to specify terminal is not a valid spoolable device. However, the spoolable characteristics of the terminal can be redefined at a later time using the $TERMUT1 utility.

END= YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

4975 Printer *(continued)*

Examples and Defaults

In the following examples, the default assignments for DEVICE=4975 (local and remote) are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignments are for illustration only.

Example 1: 4975-01R matrix printer TERMINAL statement (remotely attached).

```
label     TERMINAL  DEVICE=4975-01R,ADDRESS=4C,                   C
                LMODE=PTTOPT
```

is equivalent to:

```
label     TERMINAL  DEVICE=4975-01R,ADDRESS=4C,                    C
                PAGSIZE=66,LINSIZE=132,CHARSET=USCA,TOPM=3,        C
                BOTM=63,LEFTM=0,RIGHTM=131,OVFLINE=NO,             C
                BITRATE=1200,LMODE=PTTOPT,SPOOL=YES,END=NO
```

Note: For remote attachment, you must specify LMODE=PTTOPT as shown, in which case the BITRATE defaults to 1200. Other valid bit rates for the 4975 Models 01R and 02R are 2400 and 4800 bits per second.

Example 2: 4975-02L matrix printer TERMINAL statement (locally attached).

```
label     TERMINAL  DEVICE=4975-02L,ADDRESS=4D
```

is equivalent to:

```
label     TERMINAL  DEVICE=4975-02L,ADDRESS=4D,                      C
                PAGSIZE=66,LINSIZE=132,CHARSET=USCA,TOPM=3,          C
                BOTM=63,LEFTM=0,RIGHTM=131,OVFLINE=NO,SPOOL=YES
```

Note: The BITRATE and LMODE operands are not used for local models of the 4975 because LMODE= will always be local and BITRATE=2400 for the 4975-01L and BITRATE=4800 for the 4975-02L.

4978/4979 Display Terminals

A TERMINAL definition statement with DEVICE=4978 (or 4979) defines a 4978 display station attached via RPQ D02038 or a 4979 display station attached via 3585 adapter.

Syntax:

```
label        TERMINAL     DEVICE=,ADDRESS=,LINSIZE=,TOPM=,BOTM=,
                          NHIST=,LEFTM=,RIGHTM=,OVFLINE=,HDCOPY=,
                          ATTN=,PF1=,SCREEN=,PART=,SPOOL=,END=

Required:    DEVICE=,ADDRESS=
Defaults:    LINSIZE=80,TOPM=0,NHIST=12,BOTM=23,
             LEFTM=0,RIGHTM=79,SCREEN=ROLL,OVFLINE=NO,ATTN=YES,
             PF1=02,HDCOPY=$SYSPRTR,PART=1,SPOOL=NO,END=NO
```

Operand	*Description*
DEVICE=	One of the following device types: **4979** 4979 display station attached via 3585 Adapter **4978** 4978 display station attached via RPQ D02038
ADDRESS=	The address (in hexadecimal) of the device.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate (for example, 80 for the 4978/4979 display station), but the value cannot be increased dynamically. The default is LINSIZE=80.
TOPM=	The top margin (a decimal number between zero and the page size (24) minus 1) to indicate the top of the logical page within the physical page for the device. The default is TOPM=0.
NHIST=	The number of history lines to be retained when a page eject is performed. The line at TOPM+NHIST corresponds to logical line zero for purposes of the terminal I/O instructions. When a page eject (LINE=0) is performed, the screen area from TOPM to TOPM+NHIST-1 will contain lines from the previous page. (See the discussion of the SCREEN operand which follows.) The default is NHIST=12. **Note:** This operand is meaningful for roll screens only.
BOTM=	The bottom margin, the last usable line on a page. Its value must be between TOPM+NHIST and the page size which is 24. If an output instruction would cause the line number to increase beyond this value, then a page eject, or wrap to line zero, is performed before the operation is continued. The default is BOTM=23.

4978/4979 Display Terminals *(continued)*

LEFTM=	The left margin, the character position at which input or output will begin. Specify a decimal value between zero and LINSIZE-1. The default is LEFTM=0.
RIGHTM=	A value (between LEFTM and LINSIZE-1) that determines the last usable character position within a line. Position numbering begins at zero. The default is RIGHTM=79.
OVFLINE=	YES, if output lines which exceed the right margin are to be continued on the next line. The default is OVFLINE=NO. The overflow condition occurs when the system buffer (or a buffer in an application program) becomes full and the application program has taken no action to write the buffer to the device.
HDCOPY=	The name of the terminal to which the contents of the screen can be printed. The default is HDCOPY=$SYSPRTR.

```
HDCOPY=(terminal name,key)
```

terminal name	The symbolic name of the terminal to which the hard-copy contents will be directed.
key	The code of the program function key which is to invoke the function. For example, HDCOPY=($SYSPRTR,4) designates $SYSPRTR as the hard-copy device and PF4 as the activating key. If the hard-copy terminal name alone is specified, for example HDCOPY=$SYSPRTR, then the default is PF6.

Notes:

1. The terminal specified (terminal name) must not be defined with ATTN=NO.
2. If a terminal error occurs, for example, the printer is not turned on, your output may be lost.

ATTN= NO, if the attention key and the PF keys are to be disabled for the terminal. Such disabling is then permanent for the generated system. If you do not specify ATTN=, the default is ATTN=YES.

YES or ALL, if all attention functions are to be enabled for the terminal.

LOCAL, to limit the attention functions to those defined by ATTNLISTs within programs loaded from the terminal.

NOSYS, to exclude only the system functions ($L, $C, and so forth).

4978/4979 Display Terminals *(continued)*

NOGLOB, to exclude only the global ATTNLIST functions. (GLOBAL is the ATTNLIST of all programs in the same partition at one time.)

Note: This operand can also be entered with a two-digit hexadecimal character for the attention key if the system default is not desired.

The attention key can be redefined with a two-digit hexadecimal character.

(For more information on modifying the default attention key, refer to the description of $TERMUT2 utility in *Operator Commands and Utilities Reference.*)

Note: If the terminal being defined is specified in the HDCOPY= parameter of another terminal, do not code ATTN=NO.

PF1= The two-digit hexadecimal character which you want the system to interpret as PF key 1. The system interprets successive values as PF2 and PF3. The default is PF1=02.

SCREEN= YES or ROLL (the default), for screens that are to be operated similar to a typewriter. When the screen is full, you must press the enter key for the output to continue.

NO, for hard-copy devices. For 4978 or 4979 display devices, when the screen fills up, you do ***not*** have to press enter for the output to continue.

STATIC, for a full-screen mode of operation, if full-screen mode is supported for the device.

Note: The initial terminal definition should be STATIC only if the terminal is reserved for data display and data entry operations. Normal system operations, such as those directed to $SYSLOG or those involving the utility programs, assume a roll screen configuration. The application program can define the static screen configuration by means of the ENQT and IOCB instructions described in the *Language Reference*.

4978/4979 Display Terminals *(continued)*

PART= The partition (1–8) with which the terminal is normally associated. The default is partition 1.

You can change the partition assignment at execution time with the $CP operator command described in the *Operation Guide*.

SPOOL= YES, to define terminal as a valid spoolable device.

NO (the default), to specify terminal is not a valid spoolable device. However, the spoolable characteristics of the terminal can redefined at a later time using the $TERMUT1 utility.

END= YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

Example and Defaults

In the following example, the default assignments for the 4978/4979 display stations are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignment is for illustration only.

Example: 4978 display terminal TERMINAL statement.

```
label    TERMINAL  DEVICE=4978,ADDRESS=04
```

is equivalent to:

```
label    TERMINAL  DEVICE=4978,ADDRESS=04,LINSIZE=80,          C
                TOPM=0,NHIST=12,BOTM=23,LEFTM=0,RIGHTM=79,     C
                SCREEN=ROLL,OVFLINE=NO,ATTN=YES,PF1=02         C
                HDCOPY=$SYSPRTR,PART=1,SPOOL=NO,END=NO
```

4980 Display Station

A TERMINAL definition statement with DEVICE=4980 defines a 4980 display station attached via the Multidrop Work Station Attachment Feature (#1250).

Syntax:

```
label        TERMINAL     DEVICE=,ADDRESS=,LINSIZE=,TOPM=,BOTM=,
                          NHIST=,BITRATE=,LEFTM=,RIGHTM=,OVFLINE=,
                          HDCOPY=,ATTN=,PF1=,SCREEN=,PART=,SPOOL=,
                          ADAPTER=,PORT=,SECADDR=,END=

Required:    DEVICE=,ADDRESS=,ADAPTER=,PORT=,SECADDR=
Defaults:    LINSIZE=80,TOPM=0,BOTM=23,NHIST=12,BITRATE=100,
             LEFTM=0,RIGHTM=79,OVFLINE=NO,HDCOPY=$SYSPRTR,
             ATTN=YES,PF1=02,SCREEN=ROLL,PART=1,SPOOL=NO,END=NO
```

Operand	Description
label	A 1–8 alphanumeric character label beginning with a letter or one of the following special characters: $, #, or @.
DEVICE=	4980, for the 4980 display station.
ADDRESS=	The address (in hexadecimal) of the device.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum that the device can accommodate, but the value cannot be increased dynamically. The default is LINSIZE=80.
TOPM=	The top margin (a decimal number between zero and 23; that is, the page size minus 1) to indicate the top of the logical page within the physical page for the device. The default is TOPM=0.
NHIST=	The number of history lines to be retained when a page eject is performed. The line at TOPM+NHIST corresponds to logical line zero for purposes of the terminal I/O instructions. When a page eject (LINE=0) is performed, the screen area from TOPM to TOPM+NHIST-1 will contain pages from the previous page. (See the discussion of the SCREEN operand which follows.) The default is NHIST=12.

Note: This operand is meaningful for roll screens only.

4980 Display Station *(continued)*

BITRATE=	The line speed (in K bits per second) at which the 4980 operates. The default is 100K bits per second. The valid line speeds are 100, 250, and 500. All terminals on one port must be the same line speed and must match the line speed set on the back of the terminal. See the *4980 Display Station Description and Reference Manual*, GA21-9296, for switch settings.
BOTM=	The bottom margin, the last usable line on a page. Its value must be between TOPM+NHIST and the page size which is 24. If an output instruction causes the line number to increase beyond this value, then a page eject occurs before the operation is continued. The default is BOTM=23.
LEFTM=	The left margin, the character position at which input or output will begin. Specify a decimal value between zero and LINSIZE-1. The default is LEFTM=0.
RIGHTM=	A value (between LEFTM and LINSIZE-1) that determines the last usable character position within a line. Position numbering begins at zero. The default is RIGHTM=79.
OVFLINE=	YES, if output lines which exceed the right margin are to be continued on the next line. The default is OVFLINE=NO. The overflow condition occurs when the system buffer (or a buffer in an application program) becomes full and the application program has taken no action to write the buffer to the device.
HDCOPY=	The name of the terminal to which the contents of the screen can be printed. The default is HDCOPY=$SYSPRTR.

```
HDCOPY=(terminal name,key)
```

terminal name	The symbolic name of the terminal to which the hard-copy contents will be directed.
key	The code of the program function key which is to invoke the function. For example, HDCOPY=($SYSPRTR,4) designates $SYSPRTR as the hard-copy device and PF4 as the activating key. If you specify the name of the hard-copy terminal alone (for example HDCOPY=$SYSPRTR), then the default is PF6.

4980 Display Station *(continued)*

Notes:

1. The terminal specified (terminal name) must not be defined with ATTN=NO.
2. If a terminal error occurs (for example, if the printer is not turned on), your output may be lost.

ATTN= NO, if the attention key and the PF keys are to be disabled for the terminal. Such disabling is then permanent for the generated system.

YES (the default) or ALL, if all attention functions are to be enabled for the terminal.

LOCAL, to limit the attention functions to those defined by ATTNLISTs within programs loaded from the terminal.

NOSYS, to exclude only the system functions (such as ($L, OR $C).

NOGLOB, to exclude only the global ATTNLIST functions. (GLOBAL is the ATTNLIST of all programs in the same partition at one time.)

Note: You can also enter this operand with a two-digit hexadecimal character for the attention key if the system default is not desired.

You can redefine the attention key with a two-digit hexadecimal character.

(For more information on modifying the default attention key, refer to the description of $TERMUT2 utility in *Operator Commands and Utilities Reference.*)

Note: If the terminal being defined is specified in the HDCOPY= parameter of another terminal, do not code ATTN=NO.

PF1= The two-digit hexadecimal character that you want the system to interpret as PF key 1. The system interprets successive values as PF2 and PF3. The default is PF1=02.

SCREEN= YES or ROLL (the default), for screens that are to be operated similar to a typewriter. When the screen is full, you must press the enter key for the output to continue.

NO, for hard-copy devices. For the 4980 display station, when the screen fills up, you do ***not*** have to press enter for the output to continue.

STATIC, for a full-screen, if full-screen is supported for the device.

4980 Display Station *(continued)*

Note: The initial terminal definition should be STATIC only if you reserve the terminal for data display and data entry operations. Normal system operations, such as those directed to $SYSLOG or those involving the utility programs, assume a roll screen configuration. The application program can define the static screen configuration by means of the ENQT and IOCB instructions described in the *Language Reference*.

PART=	The partition (1–8) with which the terminal is normally associated. The default is partition 1. You can change the partition assignment at execution time with the $CP operator command described in *Operation Guide*.
SPOOL=	YES, to define terminal as a spoolable device. NO (the default), to specify terminal is not a spoolable device. However, you can redefine the spoolable characteristics of the terminal at a later time using the $TERMUT1 utility.
ADAPTER=	SMIO, for the Multidrop Work Station Attachment Feature #1250.
PORT=	The physical port on the Multidrop Work Station Attachment where the cable is connected. Its value can be 0 or 1.
SECADDR=	The address set by the switches on the back of the 4980. Each terminal connected through the same port on the SMIO must have a unique secondary address. Its value must be between 01 and FE (hex). See the *4980 Display Station Description and Reference Manual*, GA21-9296, for switch settings.
END=	YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

4980 Display Station *(continued)*

Example and Defaults

In the following example, the default assignments for the 4980 display station are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignment is for illustration only.

Example: 4980 Display Station TERMINAL statement.

```
label     TERMINAL  DEVICE=4980,ADDRESS=80,ADAPTER=SMIO,          C
                PORT=0,SECADDR=01

is equivalent to:

label     TERMINAL  DEVICE=4980,ADDRESS=80,ADAPTER=SMIO,          C
                 PORT=0,SECADDR=01,TOPM=0,NHIST=12,BITRATE=100,   C
                 BOTM=23,LEFTM=0,RIGHTM=79,SCREEN=ROLL,           C
                 OVFLINE=NO,ATTN=YES,PF1=02,PART=1,               C
                 HDCOPY=$SYSPRTR,SPOOL=NO,END=NO
```

TERMINAL - 5219/5224/5225

5219/5224/5225 Printers

A TERMINAL definition statement with DEVICE=5219/5224/5225 defines one of the following printers:

- 5219 printer
- 5224 printer
- 5225 printer.

connected to the Series/1 via the Printer Attachment - Series 5200 (#5640).

Syntax:

```
label        TERMINAL     DEVICE=,ADDRESS=,PAGSIZE=,LINSIZE=,CHARSET=,
                          TOPM=,BOTM=,LEFTM=,RIGHTM=,OVFLINE=,
                          ADAPTER=,SPOOL=,PORT=,SECADDR=,END=

 Required:   DEVICE=,ADDRESS=,ADAPTER=,PORT=,SECADDR=
 Defaults:   PAGSIZE=66,LINSIZE=132,CHARSET=USCA,TOPM=3,BOTM=63,
             LEFTM=0,RIGHTM=131,OVFLINE=NO,SPOOL=YES,END=NO
```

label	Required
Operand	***Description***
DEVICE=	One of the following device types: **5219** 5219 printer locally attached via the Printer Attachment - 5200 Series **5224** 5224 printer locally attached via the Printer Attachment - 5200 Series **5225** 5225 printer locally attached via the Printer Attachment - 5200 Series.
ADDRESS=	The address (in hexadecimal) of the device.
PAGSIZE=	The physical page size (form length) of the I/O medium. Specify a decimal number between 1 and the maximum value which is meaningful for the device. For printers, specify the number of lines per page. The default is PAGSIZE=66.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate, but the value cannot be increased dynamically. The length of an input or output line can be set at either 132 or 198. The default is LINSIZE=132.

5219/5224/5225 Printers *(continued)*

CHARSET= Select the character set for the device. Specify one of the following character sets:

AUGE	Austrian and German
BELG	Belgian
BRZL	Brazilian
DNNR	Danish and Norwegian
FRAN	French (France)
FRCA	French (Canada)
INTL	International (multinational)
ITAL	Italian
JAEN	Japanese/English
KANA	Japanese katakana
PORT	Portuguese
SPAN	Spanish (Spain)
SPNS	Spanish (countries other than Spain)
SWFI	Swedish and Finnish
UKIN	English (United Kingdom)
USCA	English (United States and Canada).

The default character set is USCA.

In addition, the characters per inch for the printers (all models) are automatically set at 10 character per inch.

These values remain in effect until you change them using either the TERMCTRL instruction or the $TERMUT1 utility.

TOPM= The top margin (a decimal number between zero and PAGSIZE-1) to indicate the top of the logical page within the physical page for the device. The default is TOPM=3.

BOTM= The bottom margin, the last usable line on a page. Its value must be between TOPM+NHIST and PAGSIZE-1. If an output instruction would cause the line number to increase beyond this value, then a page eject, or wrap to line zero, is performed before the operation is continued. The default is BOTM=63.

LEFTM= The left margin, the character position at which input or output will begin. Specify a decimal value between zero and LINSIZE-1. The default is LEFTM=0.

RIGHTM= A value (between LEFTM and LINSIZE-1) that determines the last usable character position within a line. Position numbering begins at zero. The default is RIGHTM=131.

5219/5224/5225 Printers *(continued)*

OVFLINE=	YES, if output lines which exceed the right margin are to be continued on the next line. The default is OVFLINE=NO.
	The overflow condition occurs when the system buffer (or a buffer in an application program) becomes full and the application program has taken no action to write the buffer to the device.
ADAPTER=	ALPA, for the Printer Attachment - 5200 Series (#5640). (The Printer Attachment - 5200 Series is defined with the ADAPTER definition statement.)
SPOOL=	YES, to define terminal as a valid spoolable device.
	NO, to specify terminal is not a valid spoolable device. However, the spoolable characteristics of the terminal can be redefined at a later time using the $TERMUT1 utility.
PORT=	0 for the first port, 1 for the second port.
SECADDR=	If you specify PORT=0, the second address must be 00 through 06. If you specify PORT=1, the second address must also be 00 through 06. For the 5219, this address must correspond to the address set by the switches on the back of the 5219 printer.
END=	YES for the last TERMINAL statement in a system definition module. The default is END=YES.

Examples and Defaults

In the following examples, the default assignments for DEVICE=5219/5224/5225 are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignments are for illustration only.

Example 1: 5224 printer TERMINAL statement.

```
label    TERMINAL  DEVICE=5224,ADDRESS=58,                        C
               ADAPTER=ALPA,PORT=0,SECADDR=01
```

is equivalent to:

```
label    TERMINAL  DEVICE=5224,ADDRESS=58,ADAPTER=ALPA,            C
               PAGSIZE=66,LINSIZE=132,CHARSET=USCA,TOPM=3,         C
               BOTM=63,LEFTM=0,RIGHTM=131,OVFLINE=NO,              C
               PORT=0,SECADDR=01,SPOOL=YES,END=NO
```

5219/5224/5225 Printers *(continued)*

Example 2: 5225 printer TERMINAL statement (locally attached).

```
label    TERMINAL  DEVICE=5225,ADDRESS=58,                    C
               ADAPTER=ALPA,PORT=0,SECADDR=06
```

is equivalent to:

```
label    TERMINAL  DEVICE=5225,ADDRESS=58,ADAPTER=ALPA,       C
                PAGSIZE=66,LINSIZE=132,CHARSET=USCA,TOPM=3,   C
                BOTM=63,LEFTM=0,RIGHTM=131,OVFLINE=NO,SPOOL=YES C
                PORT=0,SECADDR=06,END=YES
```

Example 3: 5219 printer TERMINAL statement (locally attached).

```
label    TERMINAL  DEVICE=5219,ADDRESS=58,                    C
               ADAPTER=ALPA,PORT=0,SECADDR=01
```

is equivalent to:

```
label    TERMINAL  DEVICE=5219,ADDRESS=58,ADAPTER=ALPA,       C
                PAGSIZE=66,LINSIZE=132,CHARSET=USCA,TOPM=3,   C
                BOTM=63,LEFTM=0,RIGHTM=131,OVFLINE=NO,        C
                SPOOL=YES,PORT=0,SECADDR=01,END=YES
```

TERMINAL - ACCA

ACCA-Type Terminals

A TERMINAL definition statement with DEVICE=ACCA means that the terminal is connected to the Series/1 using an asynchronous communications control adapter (ACCA). ACCA adapters, by feature number, are #1310, #1610, #2091 with #2092, and #2095 with #2096 or with RPQ D02350. Except for #2095 with RPQ D02350, which supports only locally-attached 3101s, these adapters support both locally-attached and remote terminals.

Syntax:

```
label       TERMINAL    DEVICE=,ADDRESS=,MODE=,PAGSIZE=,
                        LINSIZE=,CODTYPE=,TOPM=,BOTM=,NHIST=,
                        LEFTM=,RIGHTM=,OVFLINE=,LINEDEL=,
                        CHARDEL=,CRDELAY=,COD=,BITRATE=,
                        RANGE=,LMODE=,ADAPTER=,CR=,LF=,ATTN=,
                        PF1=,SCREEN=,PART=,TIMERS=,SPOOL=,END=

 Required:   DEVICE=,ADDRESS=
 Defaults:   PART=1,END=NO,OVFLINE=NO,ATTN=YES,SPOOL=NO
```

label	Required if ADAPTER=MFA; otherwise optional
Operand	***Description***
DEVICE=	ACCA, for an ASCII terminal attached via a 1610 controller, a 2091 controller with 2091 adapter, a 2095 controller with a 2096 adapter, or a 4975-01A ASCII Printer attached via a #2095/2096 or a 1310 (MFA). or ACCA, for a 3101 Display Terminal in either block or character mode attached via any of the methods above. or ACCA, for the Remote Support Link. (See "Examples and Defaults" on page IS-223 for examples.)
ADDRESS=	The address (in hexadecimal) of the device.

ACCA-Type Terminals *(continued)*

MODE= 3101C, for a 3101 Display Terminal that operates in character mode.

3101B, for a 3101 Display Terminal that operates in block mode.

4795-01A, for a 4975-01A ASCII Printer.

PAGSIZE= The physical page size (form length) of the I/O medium. Specify a decimal number between 1 and the maximum value which is meaningful for the device. For printers, specify the number of lines per page. For screen devices, specify the size of the screen in lines. This operand is not required for the 3101 Display Terminal in block mode; the system sets PAGSIZE to 24 regardless of your specification.

CODTYPE= The transmission code used by the terminal. Specify either ASCII or EBASC (8 bit data interchange code) as in the following table:

Adapter			
1610	2091/2092	2095/2096	1310 (MFA)
EBASC	EBASC	ASCII	ASCII

For example, if DEVICE=ACCA and the device is connected with a 2095 eight-line controller and a 2096 adapter, specify CODTYPE=ASCII.

Specify ASCII for a 3101 Display Terminal in either block mode or character mode for a 4975-01A Printer.

LINSIZE= The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate (for example, 80 for the 3101 display station in character mode), but the value cannot be increased dynamically.

This operand is not required for the 3101 Display Terminal in block mode; the system sets LINSIZE to 80 regardless of your specification.

TOPM= The top margin (a decimal number between zero and PAGSIZE-1) to indicate the top of the logical page within the physical page for the device.

This operand is not required for the 3101 Display Terminal in block mode; the system sets TOPM to 0 regardless of your specification.

ACCA-Type Terminals *(continued)*

NHIST=	This operand is not required for the 3101 Display Terminal in block mode; the system sets NHIST to 0 regardless of your specification. **Note:** This operand is meaningful for roll screens only.
BOTM=	The bottom margin, the last usable line on a page. Its value must be between TOPM+NHIST and PAGSIZE-1. If an output instruction causes the line number to increase beyond this value, then a page eject occurs before the operation is continued. This operand is not required for the 3101 Display Terminal in block mode; the system sets BOTM to 23 regardless of your specification.
LEFTM=	The left margin, the character position at which input or output will begin. Specify a decimal value between zero and LINSIZE-1. This operand is not required for the 3101 Display Terminal in block mode; the system sets LEFTM to 0 regardless of your specification.
RIGHTM=	A value (between LEFTM and LINSIZE-1) that determines the last usable character position within a line. Position numbering begins at zero. This operand is not required for the 3101 Display Terminal in block mode; the system sets RIGHTM to 79 regardless of your specification.
OVFLINE=	YES, if output lines which exceed the right margin are to be continued on the next line. The default is OVFLINE=NO. The overflow condition occurs when the system buffer (or a buffer in an application program) becomes full and the application program has taken no action to write the buffer to the device.
LINEDEL=	A two-digit hexadecimal character that defines the character the operator will enter when he wishes to restart an input line. In some cases, input of this character causes a repeat of the previous output message. This operand is usually not meaningful for the 3101 Display Terminal in block mode; editing is done by the device with keys such as DELETE and BACKSPACE before the edited characters are sent to the Series/1. For ACCA terminals attached via the 1610 or 2091 controllers and the 2092 adapter, code in mirror image. For information on how to code this operand for ASCII terminals, refer to "ASCII Transmission Codes" on page IS-184.

ACCA-Type Terminals *(continued)*

CHARDEL= A two-digit hexadecimal character which indicates deletion of the previous input character. It is meaningful only for devices whose mode of transmission is one character at a time, as described in the LINEDEL operand.

For ACCA terminals attached via the 1610 or 2091 controllers and the 2092 adapter, enter in mirror image. For further information on how to code this operand for ASCII terminals, refer to "ASCII Transmission Codes" on page IS-184.

Note: When CHARDEL= and LINEDEL= values are equal, no translation or editing will occur.

CRDELAY= The number of idle times required for a carriage return to complete for teletypewriter devices. If printing occurs during the carriage return, CRDELAY is too small.

BITRATE= The rate (in bits per second) that this terminal will be operating. For terminals with switch-selectable speed settings, BITRATE must match the switch setting on the terminal. (Refer to the *IBM 3101 Display Terminal Description*, GA18-2033 for information on switch settings). For best performance on the 3101 in block mode, specify 9600 if you do not use modems or the maximum speed capacity of the modem you use.

For the 4975-01A ASCII Printer attached via a Feature Programmable Card (#2095/2096), specify BITRATE=1200 or 4800. If you attach the printer via a Multifunction Attachment Feature #1310, specify BITRATE=1200 only.

For ACCA terminals, you may specify BITRATE=0. You should only specify this bit rate if the terminal is attached via the #2095 controller and the #2096 adapter and uses the external clock.

Notes:

1. The MFA (#1310) can run only at 1200 bits per second for the 4975-01A ASCII Printer and 1200, 2400, and 9600 bits per second for other terminals.
2. For the Remote Support Link, specify BITRATE=1200. (See "Examples and Defaults" on page IS-223 for examples.)

Default BITRATE is 300 bits per second; the default for MFA is 1200 bits per second.

ACCA-Type Terminals *(continued)*

RANGE=	Enter HIGH or LOW to match hardware jumper that is installed on the adapter card. The Multifunction Attachment does not have a speed range jumper. On the 3101 Display Terminal in block mode, specify HIGH for best performance. Default RANGE is HIGH.
LMODE=	For ACCA devices, four options are available: RS-422, LOCAL, SWITCHED, or PTTOPT.

RS-422	For a terminal directly attached to any port of the Multifunction Attachment Feature #1310.
LOCAL	For a terminal directly attached. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only.
SWITCHED	For a connection that is point-to-point switched. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only.
PTTOPT	For a connection that is point-to-point nonswitched. For the the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only.

The LMODE specified should match the jumpers on the controller cards. The Multifunction Attachment does not have jumpers. (Refer to Appendix B, "Customizing Adapters with Hardware Jumpers" on page IS-243 for hardware considerations.)

Note: Specify LMODE=SWITCHED for the Remote Support Link. (See "Examples and Defaults" on page IS-223 for examples.)

ACCA-Type Terminals *(continued)*

ADAPTER= One of the following to indicate the ACCA type:

SINGLE	For a single-line controller (the default)
TWO	For an eight-line controller with up to two lines active
FOUR	For an eight-line controller with up to four lines active
SIX	For an eight-line controller with up to six lines active
EIGHT	For an eight-line controller with up to eight lines active
MFA	For a Multifunction Attachment Feature #1310.

All multiple-line asynchronous controller/adapter cards (2091/2092 or 2095/2096) must start at a base address ending with X'0' or X'8'. A terminal statement with DEVICE=ACCA must exist for the line at the base address. Furthermore, the terminal defined as the base address must be specified as the first terminal for the multiline controller. The remaining terminals defined on the multiline controller (***if any***) must immediately follow the base address terminal and should be in ascending order by address.

All 3101 terminals attached via the Multifunction Attachment (1310) must specify ADAPTER=MFA.

COD= Additional characters, other than the CR=, ATTN=, and LINEDEL= values, that will terminate a READTEXT operation. You may enter from one to four characters. (COD means change of direction, for example, read to write.) When the system receives any COD characters, it passes them to the READTEXT data area. Code in ASCII mirror image as follows (depending upon device type):

```
COD=11
or
COD=(12,B6,42,B3,..)
```

CR= The character to be tested to determine if a new line function is to be performed. When the system receives any CR characters, it does not pass them to the READTEXT data area. (Code in mirror image for ACCA terminals attached via the 1610 or 2091 controllers with the 2092 adapter.)

Default CR is B0 for EBASC and 0D for ASCII.

ACCA-Type Terminals *(continued)*

LF=	The character to be sent to the terminal when a new line function is to be performed. Code in mirror image for ACCA terminals attached via the 1610 or 2091 controllers with the 2092 adapter. If the same value is coded for LF= as was coded (or defaulted) for CR= then the CR character which terminates an input operation will not be echoed to the terminal; the terminal is assumed to be an auto line-feed device. Default LF is 50 for EBASC and 0A for ASCII.
ATTN=	NO, if the attention key and the PF keys are to be disabled for the terminal. Such disabling is then permanent for the generated system. YES (the default) or ALL, if all attention functions are to be enabled for the terminal. LOCAL, to limit the attention functions to those defined by ATTNLISTs within programs loaded from the terminal. NOSYS, to exclude only the system functions ($L, $C, and so forth). NOGLOB, to exclude only the global ATTNLIST functions. (GLOBAL is the ATTNLIST of all programs in the same partition at one time.) **Note:** This operand can also be entered with a two-digit hexadecimal character for the attention key if the system default is not desired. The attention key can be redefined with a two-digit hexadecimal character for ASCII terminals. For terminals attached via the 1610 or 2091 controllers and the 2092 adapter, use mirror image. For the 3101 Display Terminal in either character or block mode, the system defines the PF8 key as the default attention key. If you do not want the system default, specify two characters (four hexadecimal digits) representing the first and second characters of a 3101 PF key sequence. If only one character (two hexadecimal digits) is coded, the second character defaults to binary zeroes. (For information on 3101 PF key sequences, refer to *IBM 3101 Display Terminal Description*, GA18-2033.) For terminals other than the 3101 Display Terminal, the ASCII default is X'1B' (the ESC key). The EBASC default is X'D8'.

ACCA-Type Terminals *(continued)*

PF1=	The two-digit hexadecimal character which you want the system to interpret as PF key 1. The system interprets successive values as PF2 and PF3.

For the 3101 Display Terminal, defaults are listed in "Examples and Defaults" on page IS-223.

For the 3101 Display Terminal in either character or block mode, the system defines the PF1 key as the default PF1 key. If you do not want the system default, specify two characters (four hexadecimal digits) representing the first and second characters of a 3101 PF key sequence. The system interprets successive characters as PF2, PF3, ... PF8. If you code only one character (two hexadecimal digits), the system makes the second character binary zeroes.

For a 3101 Display Terminal attached via a 1610 controller or a 2091 controller with a 2092 adapter, use mirror image.

The PF1 ASCII default is 1B61; the EBASC default is D886.

SCREEN= YES or ROLL, for screens that are to be operated similar to a typewriter. For screen devices that are attached through the teletypewriter adapter, when the screen is full, you must press the enter key for the output to continue.

NO, for hard-copy devices. For 3101 display devices, when the screen fills up, you do ***not*** have to press enter for the output to continue.

STATIC, for a full-screen mode of operation. STATIC is valid only for the 3101 Display Terminal in block mode.

Note: The initial terminal definition should be STATIC only if the terminal is reserved for data display and data entry operations. Normal system operations, such as those directed to $SYSLOG or those involving the utility programs, assume a roll screen configuration. The application program can define the static screen configuration by means of the ENQT and IOCB instructions described in the *Language Reference*.

ACCA-Type Terminals *(continued)*

PART=	The partition (1–8) with which the terminal is normally associated. The default is partition 1. You can change the partition assignment at execution time with the $CP operator command described in *Operation Guide*.
TIMERS=	The ACCA T1 and T2 values for receive and transmit operations in a sublist format. The timer defaults depend on the LMODE specified. For LOCAL and RS-422, the defaults are 1,1,1,1. For PTTOPT, the defaults are 2,2,10,2 and for SWITCHED, they are 2,2,10,300. For example, TIMERS=(4,5,6,7) specifies:

Receive T1	= 4
Receive T2	= 5
Transmit T1	= 6
Transmit T2	= 7.

SPOOL=	YES, to define terminal as a valid spoolable device. NO (the default), to specify terminal is not a valid spoolable device. However, the spoolable characteristics of the terminal can be redefined at a later time using the $TERMUT1 utility.
END=	YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

ACCA-Type Terminals *(continued)*

Examples and Defaults

Default values for optional parameters on the TERMINAL statement vary with the device type. In the following examples, the default assignments for each device support are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignments are for illustration only.

Example 1: ASCII terminal via 1610 controller TERMINAL statement.

```
label     TERMINAL  DEVICE=ACCA,ADDRESS=70,PAGSIZE=35,        C
               LINSIZE=80,CODTYPE=EBASC,TOPM=0,BOTM=34,       C
               LEFTM=0,RIGHTM=79,SCREEN=NO,OVFLINE=NO,        C
               CRDELAY=0,BITRATE=300,RANGE=HIGH,              C
               LMODE=PTTOPT,ATTN=YES,ADAPTER=SINGLE,LF=50,    C
               CHARDEL=10,LINEDEL=FE,CR=B0,PART=1,            C
               TIMERS=(2,2,10,2),SPOOL=NO,END=NO
```

The following examples contain the required operands and the default values for defining the 3101 Display Terminal. Address assignments are for illustration only. The default values require that you set space parity in the parity bit selection setup switches. (Refer to Appendix C, "3101 Configuration Information" on page IS-249 and *IBM 3101 Display Terminal Description*, GA18-2033 information on switch settings.)

Example 2: IBM 3101 in character mode (connected with 1610 single-line controller or 2091 eight-line controller with 2092 four-line adapter).

```
label     TERMINAL  DEVICE=ACCA,ADDRESS=08,MODE=3101C
```

is equivalent to:

```
label    TERMINAL  DEVICE=ACCA,ADDRESS=08,MODE=3101C,           C
               PAGSIZE=35,LINSIZE=80,CODTYPE=EBASC,TOPM=0,      C
               BOTM=34,LEFTM=0,RIGHTM=79,SCREEN=NO,NHIST=0,     C
               OVFLINE=NO,CRDELAY=0,BITRATE=300,RANGE=HIGH,     C
               LMODE=PTTOPT,ATTN=D816,ADAPTER=SINGLE,LF=50,     C
               CHARDEL=10,LINEDEL=FE,CR=B0,PF1=D886,            C
               TIMERS=(2,2,10,2),SPOOL=NO
```

ACCA-Type Terminals *(continued)*

Example 3: IBM 3101 in block mode (connected with 1610 single-line controller or 2091 eight-line controller with 2092 four-line adapter).

```
label     TERMINAL   DEVICE=ACCA,ADDRESS=68,MODE=3101B

is equivalent to:

label     TERMINAL   DEVICE=ACCA,ADDRESS=68,MODE=3101B,PAGSIZE=24, C
                LINSIZE=80,CODTYPE=EBASC,TOPM=0,BOTM=23,LEFTM=0,  C
                RIGHTM=79,SCREEN=ROLL,NHIST=0,OVFLINE=NO,         C
                CRDELAY=0,BITRATE=300,RANGE=HIGH,LMODE=PTTOPT,    C
                ATTN=D816,ADAPTER=SINGLE,LF=50,CR=B0,PF1=D886,    C
                TIMERS=(2,2,10,2),SPOOL=NO
```

Note: When DEVICE=ACCA and you code the MODE operand, the system assumes additional defaults relevant to the 3101 Display Terminal.

The following two examples contain sample values for defining the 3101 Display Terminal when it is connected with the 2095 controller and 2096 adapter. These examples are intended to guide you in coding your TERMINAL statements.

The values shown require that you set space parity in the parity bit selection setup switches (refer to the *IBM 3101 Display Terminal Description*, GA18-2033, for information on switch settings). See Appendix C, "3101 Configuration Information" on page IS-249 for recommended switch settings for the 3101 Display Terminal. Address assignments are for illustration only.

Example 4: IBM 3101 in character mode (connected with 2095 eight-line controller and 2096 four-line adapter).

```
label     TERMINAL   DEVICE=ACCA,ADDRESS=60,MODE=3101C,              C
                PAGSIZE=35,LINSIZE=80,CODTYPE=ASCII,TOPM=0,          C
                BOTM=34,LEFTM=0,RIGHTM=79,SCREEN=NO,NHIST=0,         C
                OVFLINE=NO,CRDELAY=0,BITRATE=300,RANGE=HIGH,         C
                LMODE=PTTOPT,ATTN=1B68,ADAPTER=FOUR,LF=0A,           C
                CHARDEL=08,LINEDEL=7F,CR=0D,PF1=1B61,                C
                TIMERS=(2,2,10,2),SPOOL=NO
```

ACCA-Type Terminals *(continued)*

Example 5: IBM 3101 in block mode (connected with 2095 eight-line controller and 2096 four-line adapter).

```
label     TERMINAL  DEVICE=ACCA,ADDRESS=61,MODE=3101B,              C
                 PAGSIZE=24,LINSIZE=80,CODTYPE=ASCII,               C
                 TOPM=0,BOTM=23,LEFTM=0,RIGHTM=79,                  C
                 SCREEN=ROLL,NHIST=0,OVFLINE=NO,CRDELAY=0,          C
                 BITRATE=300,RANGE=HIGH,LMODE=PTTOPT,               C
                 ATTN=1B68,ADAPTER=FOUR,LF=0A,CR=0D,PF1=1B61,       C
                 TIMERS=(2,2,10,2),SPOOL=NO
```

The following two examples contain sample values for defining the 3101 Display Terminal when it is connected with the #1310 adapter. These examples are intended to guide you in coding your TERMINAL statement.

Example 6: IBM 3101 in character mode (connected to a #1310 (MFA) base address).

```
label     TERMINAL  DEVICE=ACCA,ADDRESS=58,MODE=3101C,            C
                 PAGSIZE=35,LINSIZE=80,CODTYPE=ASCII,             C
                 TOPM=0,BOTM=34,LEFTM=0,RIGHTM=79,                C
                 SCREEN=ROLL,NHIST=0,OVFLINE=NO,CRDELAY=0,        C
                 BITRATE=1200,LMODE=PTTOPT,ATTN=1B68,LF=0A,       C
                 CHARDEL=08,LINEDEL=7F,CR=0D,PF1=1B61,            C
                 ADAPTER=MFA,PART=1,TIMERS=(2,2,10,2),            C
                 SPOOL=NO,END=NO
```

Example 7: IBM 3101 in block mode (connected to a #1310 (MFA) that is not the base address).

```
label     TERMINAL  DEVICE=ACCA,ADDRESS=59,MODE=3101B,            C
                 PAGSIZE=24,LINSIZE=80,CODTYPE=ASCII,             C
                 TOPM=0,BOTM=23,LEFTM=0,RIGHTM=79,                C
                 SCREEN=ROLL,NHIST=0,OVFLINE=NO,CRDELAY=0,        C
                 BITRATE=1200,LMODE=RS422,ATTN=1B68,LF=0A,        C
                 CR=0D,PF1=1B61,ADAPTER=MFA,                      C
                 TIMERS=(1,1,1,1),SPOOL=NO,END=NO
```

ACCA-Type Terminals *(continued)*

Example 8: The following example contains sample values for defining the 4975-01A Printer when it is connected via the Feature Programmable Attachment (2095/2096). You must code ADAPTER=MFA if you use MFA to attach a 4975-01A Printer.

```
label    TERMINAL  DEVICE=ACCA,ADDRESS=61,BITRATE=1200,        C
               LMODE=PTTOPT,ADAPTER=FOUR,TOPM=3,BOTM=62,       C
               LINSIZE=132,RANGE=HIGH,CODTYPE=ASCII,           C
               PAGSIZE=66,MODE=4975-01A,COD=(11,13,1B),        C
               TIMERS=(12,2,5,2)
```

The Remote Support Link

The following examples contain the required operands for defining the Remote Support Link for the four different types of attachments. You ***must*** define the Remote Support Link as an ACCA terminal on a switched line. (Refer to the *Problem Determination Guide* for information on using the Remote Support Link.)

Example 9: With a 1610 attachment.

```
label    TERMINAL  DEVICE=ACCA,ADDRESS=08,BITRATE=1200,        C
               LMODE=SWITCHED,ADAPTER=SINGLE,PF1=D886,         C
               CODTYPE=EBASC,ATTN=D816,PAGSIZE=24,PART=3,      C
               SCREEN=YES
```

Example 10: With a 2095/2096 attachment.

```
label    TERMINAL  DEVICE=ACCA,ADDRESS=70,BITRATE=1200,        C
               LMODE=SWITCHED,ADAPTER=FOUR,PF1=1B61,           C
               CODTYPE=ASCII,ATTN=1B68,PAGSIZE=24,PART=3,      C
               SCREEN=YES
```

Example 11: With an MFA attachment.

```
label    TERMINAL  DEVICE=ACCA,ADDRESS=58,BITRATE=1200,        C
               LMODE=SWITCHED,ADAPTER=MFA,PF1=1B61,            C
               CODTYPE=ASCII,ATTN=1B68,PAGSIZE=24,PART=3,      C
               SCREEN=YES
```

ACCA-Type Terminals *(continued)*

Example 12: With a 2091/2092 attachment.

```
label    TERMINAL  DEVICE=ACCA,ADDRESS=70,BITRATE=1200,       C
               LMODE=SWITCHED,ADAPTER=FOUR,PF1=D886,          C
               CODTYPE=EBASC,ATTN=D816,PAGSIZE=24,PART=3,     C
               SCREEN=YES
```

TTY-Type Terminals

A TERMINAL definition statement with DEVICE=TTY locally attaches teletypewriter terminals and IBM 3101 Display Terminals to the Series/1 through the Teletypewriter Attachment (feature #7850). Its most frequent use is in transferring ASCII character strings between the Series/1 and a teletypewriter terminal. The most common types of such terminals are keyboard/printer and keyboard/CRT devices.

Devices that may be compatible with the physical transmission methods of the Series/1 Teletypewriter Adapter are available from many vendors; these include Isolated Contact sense, TTL, and EIA. Such devices include terminals that transmit only, or receive only, or transmit only in response to being polled. The devices may not have keyboards for input but may acquire data from bar code scanners or analog or digital input features within the device. The transmission code used by these devices can be alphameric ASCII or any of the 256 possible 8-bit character combinations.

Syntax:

```
label      TERMINAL     DEVICE=,ADDRESS=,MODE=,PAGSIZE=,CODTYPE=,
                        LINSIZE=,TOPM=,BOTM=,LEFTM,RIGHTM=,OVFLINE=,
                        LINEDEL=,CHARDEL=,CRDELAY=,ECHO=,CR=,
                        LF=,ATTN=,PF1=,SCREEN=,PART=,SPOOL=,END=

Required:  DEVICE=,ADDRESS=
Defaults:  OVFLINE=NO,ECHO=YES,CR=0D,LF=0A,ATTN=YES,
           SPOOL=NO,PART=1,END=NO
```

Operand	***Description***
DEVICE=	TTY, a 3101 Display Terminal in character mode or another ASCII Terminal attached via Teletypewriter Adapter (7850)
ADDRESS=	The address (in hexadecimal) of the device.
MODE=	3101C, for a 3101 Display Terminal that operates in character mode.
PAGSIZE=	The physical page size (form length) of the I/O medium. Specify a decimal number between 1 and the maximum value which is meaningful for the device.
CODTYPE=	The transmission code used by the terminal; in this case, CODTYPE= must equal ASCII.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate, but the value cannot be increased dynamically.
TOPM=	The top margin (a decimal number between zero and PAGSIZE-1) to indicate the top of the logical page within the physical page for the device.

TTY-Type Terminals *(continued)*

BOTM=	The bottom margin, the last usable line on a page. Its value must be between TOPM+NHIST and PAGSIZE-1. If an output instruction would cause the line number to increase beyond this value, then a page eject, or wrap to line zero, is performed before the operation is continued.
LEFTM=	The left margin, the character position at which input or output will begin. Specify a decimal value between zero and LINSIZE-1.
RIGHTM=	A value (between LEFTM and LINSIZE-1) that determines the last usable character position within a line. Position numbering begins at zero.
OVFLINE=	YES, if output lines which exceed the right margin are to be continued on the next line. The default is OVFLINE=NO. The overflow condition occurs when the system buffer (or a buffer in an application program) becomes full and the application program has taken no action to write the buffer to the device.
LINEDEL=	A two-digit hexadecimal character that defines the character the operator will enter when he wishes to restart an input line. In some cases, input of this character causes a repeat of the previous output message. Refer to "ASCII Transmission Codes" on page IS-184.
CHARDEL=	A two-digit hexadecimal character which indicates deletion of the previous input character. It is meaningful only for devices whose mode of transmission is one character at a time, as described in the LINEDEL operand. For further information on how to code this operand for ASCII terminals, refer to "ASCII Transmission Codes" on page IS-184. **Note:** When CHARDEL= and LINEDEL= values are equal, no translation or editing will occur.
CRDELAY=	The number of idle times required for a carriage return to complete for teletypewriter devices. If printing occurs during the carriage return, CRDELAY is too small.
ECHO=	NO, for devices that do not require input characters to be written back (echoed) by the processor for printing. YES (the default) is appropriate for most devices connected through the teletypewriter adapter. See the LF parameter description regarding suppression of the echo of the CR character.

TTY-Type Terminals *(continued)*

CR=	The character to be tested to determine if a new line function is to be performed. The default is CR=0D. For further information on how to code this operand for ASCII terminals, refer to "ASCII Transmission Codes" on page IS-184.
LF=	The character to be sent to the terminal when a new line function is to be performed. If the same value is coded for LF= as was coded (or defaulted) for CR= then the CR character which terminates an input operation will not be echoed to the terminal; the terminal is assumed to be an auto line-feed device. The default is LF=0A. For further information on how to code this operand for ASCII terminals, refer to "ASCII Transmission Codes" on page IS-184.
ATTN=	NO, if the attention key and the PF keys are to be disabled for the terminal. Such disabling is then permanent for the generated system.

YES (the default) or ALL, if all attention functions are to be enabled for the terminal.

LOCAL, to limit the attention functions to those defined by ATTNLISTs within programs loaded from the terminal.

NOSYS, to exclude only the system functions ($L, $C, and so forth).

NOGLOB, to exclude only the global ATTNLIST functions. (GLOBAL is the ATTNLIST of all programs in the same partition at one time.)

Note: This operand can also be entered with a two-digit hexadecimal character for the attention key if the system default is not desired.

The attention key can be redefined with a two-digit hexadecimal character for ASCII terminals.

For the 3101 Display Terminal in character mode, the system defines the PF8 key as the default attention key. If you do not want the system default, specify two characters (four hexadecimal digits) representing the first and second characters of a 3101 PF key sequence. If only one character (two hexadecimal digits) is coded, the second character defaults to binary zeroes.

Note: The attention key must be uniquely defined so that the hexadecimal digits cannot be returned as the response to a READTEXT statement. If the key is not uniquely defined, the attention routine is entered erroneously.

TTY-Type Terminals *(continued)*

(For more information on modifying the default attention key, refer to the description of $TERMUT2 utility in *Operator Commands and Utilities Reference*. For information on 3101 PF key sequences, refer to *IBM 3101 Display Terminal Description*, GA18-2033.)

PF1= The two-digit hexadecimal character which you want the system to interpret as PF key 1. The system interprets successive values as PF2 and PF3.

For the 3101 Display Terminal, defaults are listed in "Examples and Defaults" on page IS-232.

For the 3101 Display Terminal in character mode, the system defines the PF1 key as the default PF1 key. If you do not want the system default, specify two characters (four hexadecimal digits) representing the first and second characters of a 3101 PF key sequence. The system interprets successive characters as PF2, PF3, ... PF8. If you code only one character (two hexadecimal digits), the system makes the second character binary zeroes.

SCREEN= YES or ROLL, for screens that are to be operated similar to a typewriter. For screen devices that are attached through the teletypewriter adapter, when the screen is full, you must press the enter key for the output to continue.

NO, for 3101 display devices. When the screen fills up, you do ***not*** have to press enter for the output to continue.

PART= The partition (1–8) with which the terminal is normally associated. The default is partition 1.

You can change the partition assignment at execution time with the $CP operator command described in *Operation Guide*.

SPOOL= YES, to define terminal as a valid spoolable device.

NO (the default), to specify terminal is not a valid spoolable device. However, the spoolable characteristics of the terminal can be redefined at a later time using the $TERMUT1 utility.

END= YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

TTY-Type Terminals *(continued)*

Examples and Defaults

In the following examples, the default assignments for DEVICE=TTY are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignments are for illustration only.

Example 1: ASCII terminal via 7850 adapter TERMINAL statement.

```
label     TERMINAL  DEVICE=TTY,ADDRESS=00,PAGSIZE=35,LINSIZE=80,  C
                CODTYPE=ASCII,TOPM=0,BOTM=34,LEFTM=0,RIGHTM=79,   C
                SCREEN=NO,OVFLINE=NO,LINEDEL=7F,CHARDEL=08,       C
                CRDELAY=0,ECHO=YES,ATTN=YES,CR=0D,LF=0A,SPOOL=NO
```

The following example contains the required operands and the default values for defining the 3101 Display Terminal in character mode. The default values require that you set space parity in the parity bit selection setup switches. (Refer to *IBM 3101 Display Terminal Description*, GA18-2033 and Appendix C, "3101 Configuration Information" on page IS-249 for information on switch settings.)

Example 2: IBM 3101 in character mode (connected with 7850 teletypewriter adapter).

```
label     TERMINAL  DEVICE=TTY,ADDRESS=00,MODE=3101C
```

is equivalent to:

```
label     TERMINAL  DEVICE=TTY,ADDRESS=00,MODE=3101C,PAGSIZE=35,  C
                LINSIZE=80,CODTYPE=ASCII,TOPM=0,BOTM=34,          C
                LEFTM=0,RIGHTM=79,SCREEN=NO,NHIST=0,OVFLINE=NO,   C
                LINEDEL=7F,CHARDEL=08,CRDELAY=0,ECHO=YES,         C
                ATTN=1B68,CR=0D,LF=0A,PF1=1B61,SPOOL=NO
```

Note: When DEVICE=TTY, if you code the MODE operand, the system assumes additional defaults relevant to the 3101 Display Terminal.

Processor-to-Processor

A TERMINAL definition statement with DEVICE=PROC is used when the Series/1 is to communicate with another processor. The Asynchronous Communication Single Line Controller (feature #1610) is the feature used. This allows connecting Series/1-to-Series/1, or Series/1 to any other processor capable of handling the required protocols. As with terminals, ATTENTION signals can be transmitted.

If CODTYPE=EBCDIC is defined on the TERMINAL statement, arbitrary binary data can be transmitted. Set the BITRATE and RANGE parameters in accordance with the hardware jumpers, matching the setting in the other processor. Also, the LINSIZE parameter must have the same value in both processors.

The transmission protocol can be modified to satisfy special requirements by assigning the appropriate values to the CRDELAY and CODTYPE operands.

Syntax:

```
label        TERMINAL     DEVICE=,ADDRESS=,CODTYPE=,
                          LINSIZE=,CRDELAY=,BITRATE=,
                          CR=,LF=,RANGE=,END=

Required:    DEVICE=,ADDRESS=
Defaults:    CODTYPE=EBCDIC,LINSIZE=130,CRDELAY=(PROMPT,30000),
             BITRATE=9600,RANGE=HIGH,CR=5B,LF=5B,END=NO
```

Operand	*Description*
DEVICE=	PROC, processor-to-processor communication.
ADDRESS=	The address (in hexadecimal) of the device.
CODTYPE=	The transmission code used by the terminal. Specify either CRSP, EBCD, or EBCDIC as follows. The default is CODTYPE=EBCDIC.

CRSP	With this option, the #1610 controller is set to PTTC mode (see *Communications Feature Description*) and messages are translated using the CRSP conversion table (PTTC/correspondence code). The communication is restricted to characters, as PTTC mode allows only the transmission of bytes with the seven low-order bits of odd parity. Therefore, XLATE=NO should not be specified on PRINTEXT or READTEXT instructions.

Processor-to-Processor *(continued)*

EBCD The effect of coding this option is similar to CRSP, except that the EBCD conversion table is used. The EBCD option is recommended for connection to an IBM 5100 or 5110 computer. The 6-bit code must be selected with the Serial I/O feature.

EBCDIC This option sets the #1610 controller to Eight Bit Coded Data Interchange mode with all change of direction codes equal to X'FF' (see the *Communications Feature Description*). Special protocol provides for transparent exchange of arbitrary binary data. As there are no parity restrictions and only the code X'FF' is recognized as change of direction (indicating EOT, NL or EOSR), all bytes (especially all EBCDIC characters) other than X'FF' are transmitted "as is." Before a message or record is sent, it is scanned for a byte code (other than X'FF') not contained in it. This special code is sent as EOA and every occurring X'FF' in the message or record is replaced by it. On the receiving side, every EOA code is replaced by X'FF'. If a record is larger than 128 bytes, it is divided into appropriate subrecords (length < 128 bytes) to which the procedure can be applied.

LINSIZE= The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate, but the value cannot be increased dynamically. The default is LINSIZE=130.

CRDELAY= The number of idle times required for a carriage return to complete for teletypewriter devices. If printing occurs during the carriage return, CRDELAY is too small. The default is CRDELAY=PROMPT,30000.

PROMPT,n The device support waits before every record (and subrecord) for the EOT prompt character. The time limit is n times 3.33 milliseconds, starting at the end of the previous operation. In response to the EOT, and also at the beginning of every record (and subrecord), an EOA character is sent.

SP5100,n The effect of coding this option is identical to the PROMPT mode except that at End of Record, the two characters Line Feed and New Line (X'3B5B') are sent. This is necessary for communication with the IBM 5100 or 5110 running APL or BASIC and using the Serial I/O feature.

DELAY,n At the beginning of a message, the device support waits a maximum of one second for the EOT character(s). After each record, a delay of n times 3.33 milliseconds is inserted. This mode might be used to simulate an 2741-like terminal for another processor.

Processor-to-Processor *(continued)*

Note: When CHARDEL= and LINEDEL= values are equal, no translation or editing will occur.

BITRATE= The rate (in bits per second) that this terminal will be operating. The default is BITRATE=9600.

RANGE= Enter HIGH or LOW to match hardware jumper that is installed on the adapter card. The default is RANGE=HIGH.

CR= The character to be tested to determine if a new line function is to be performed.

The default is CR=5B.

LF= The character to be sent to the terminal when a new line function is to be performed. If the same value is coded for LF= as was coded (or defaulted) for CR= then the CR character which terminates an input operation will not be echoed to the terminal; the terminal is assumed to be an auto line-feed device.

The default is LF=5B.

END= YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

Example and Defaults

In the following example, the default assignments for DEVICE=PROC are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for this support. Address assignment is for illustration only.

Example: Defining processor-to-processor communications through the 1610 controller.

```
label     TERMINAL   DEVICE=PROC,ADDRESS=7F

is equivalent to:

label     TERMINAL   DEVICE=PROC,ADDRESS=7F,CODTYPE=EBCDIC,          C
                 LINSIZE=130,CRDELAY=(PROMPT,30000),BITRATE=9600,    C
                 RANGE=HIGH,CR=5B,LF=5B,END=NO
```

TERMINAL - Virtual Terminal

Interprogram Communication - Virtual Terminals

A TERMINAL definition statement with DEVICE=VIRT defines interprogram communication. Refer to the *Event Driven Executive Language Programming Guide* for a description of virtual terminals.

Syntax:

```
label        TERMINAL     DEVICE=,ADDRESS=,LINSIZE=,
                          SYNC=,END=

Required:    DEVICE=,ADDRESS=
Defaults:    LINSIZE=80,SYNC=NO,END=NO
```

Operand	***Description***
DEVICE=	VIRT for interprogram communication.
ADDRESS=	The label of the other virtual terminal.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate, but the value cannot be increased dynamically. The default is LINSIZE=80.
SYNC=	YES, if synchronization events will be posted to this virtual terminal. Attempted actions over the virtual channel are indicated in the task control word. YES allows two terminals to synchronize their actions so that when one terminal is writing, the other is reading.
	NO, if you do not want synchronization events posted. NO is the default.
END=	YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

Interprogram Communication - Virtual Terminals *(continued)*

Examples and Defaults

In the following examples, the default assignments for DEVICE=VIRT are shown as if they were explicitly coded in the TERMINAL statement. If a parameter is not shown, then it is not relevant for the device. Address assignments are for illustration only.

Example 1: Remote Management Utility using the PASSTHRU function - TERMINAL statements.

Note: This example shows a line size of 80. The maximum line size value is 254. The names CDRVTA and CDRVTB are required.

```
CDRVTA    TERMINAL   DEVICE=VIRT,ADDRESS=CDRVTB,SYNC=YES

CDRVTB    TERMINAL   DEVICE=VIRT,ADDRESS=CDRVTA

is equivalent to:

CDRVTA    TERMINAL   DEVICE=VIRT,ADDRESS=CDRVTB,SYNC=YES,    C
                LINSIZE=80,END=NO

CDRVTB    TERMINAL   DEVICE=VIRT,ADDRESS=CDRVTA,SYNC=NO,     C
                LINSIZE=80,END=NO
```

Example 2: Virtual terminal TERMINAL statements.

```
AAA         TERMINAL   DEVICE=VIRT,ADDRESS=BBB,SYNC=YES

BBB         TERMINAL   DEVICE=VIRT,ADDRESS=AAA

is equivalent to:

AAA         TERMINAL   DEVICE=VIRT,ADDRESS=BBB,LINSIZE=80,    C
                 SYNC=YES,END=NO

BBB         TERMINAL   DEVICE=VIRT,ADDRESS=AAA,LINSIZE=80,    C
                 SYNC=NO,END=NO
```

TERMINAL - GPIB

General Purpose Interface Bus

A TERMINAL definition statement with DEVICE=GPIB defines a General Purpose Interface Bus (GPIB). Refer to *Communications Guide* for a description of GPIB.

Syntax:

```
 label      TERMINAL     DEVICE=,ADDRESS=,LINSIZE=,CODTYPE=,
                         ATTN=,SCREEN=,PART=,SPOOL=,END=

Required:   DEVICE=,ADDRESS=
Defaults:   CODTYPE=ASCII,LINSIZE=80,ATTN=YES,SCREEN=NO,
            PART=1,SPOOL=NO,END=NO
```

Operand	***Description***
DEVICE=	GPIB, General Purpose Interface Bus (GPIB).
ADDRESS=	The address (in hexadecimal) of the device.
CODTYPE=	The transmission code used by the terminal. In this case, specify ASCII.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate, but the value cannot be increased dynamically. The default is LINSIZE=80.
ATTN=	NO, if the attention key and the PF keys are to be disabled for the terminal. Such disabling is then permanent for the generated system. If you do not specify ATTN=, the default is ATTN=YES.

YES or ALL, if all attention functions are to be enabled for the terminal.

LOCAL, to limit the attention functions to those defined by ATTNLISTs within programs loaded from the terminal.

NOSYS, to exclude only the system functions ($L, $C, and so forth).

NOGLOB, to exclude only the global ATTNLIST functions. (GLOBAL is the ATTNLIST of all programs in the same partition at one time.)

Note: This operand can also be entered with a two-digit hexadecimal character for the attention key if the system default is not desired.

The attention key can be redefined with a two-digit hexadecimal character for ASCII terminals.

For ASCII terminals other than the 3101 Display Terminal, the default is ASCII X'1B' (the ESC key). The mirror image of X'1B' is X'D8'.

General Purpose Interface Bus *(continued)*

Note: If the terminal being defined is specified in the HDCOPY= parameter of another terminal, do not code ATTN=NO.

SCREEN= YES or ROLL, for screens that are to be operated similar to a typewriter. For screen devices that are attached through the teletypewriter adapter, when the screen is full, you must press the enter key for the output to continue.

NO (the default), for hard-copy devices. When the screen fills up, you do ***not*** have to press enter for the output to continue.

PART= The partition (1–8) with which the terminal is normally associated. The default is partition 1.

You can change the partition assignment at execution time with the $CP operator command described in *Operation Guide*.

SPOOL= YES, to define terminal as a valid spoolable device.

NO (the default), to specify terminal is not a valid spoolable device. However, the spoolable characteristics of the terminal can be redefined at a later time using the $TERMUT1 utility.

END= YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

Example and Defaults

In the following example, the default assignments for DEVICE=GPIB are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignment is for illustration only.

Example: General Purpose Interface Bus (GPIB) TERMINAL statement.

```
label     TERMINAL  DEVICE=GPIB,ADDRESS=25
```

is equivalent to:

```
label     TERMINAL  DEVICE=GPIB,ADDRESS=25,LINSIZE=80,      C
                CODTYPE=ASCII,SCREEN=NO,ATTN=YES,PART=1,    C
                SPOOL=NO,END=NO
```

Series/1-to-Series/1

A TERMINAL definition statement with DEVICE=S1S1 defines the Series/1-to-Series/1 Attachment (RPQ D02241 or D02242). Refer to *Communications Guide* for a description of the attachment.

Syntax:

```
label        TERMINAL     DEVICE=,ADDRESS=,LINSIZE=,
                          CHKSUM=,END=

Required:    DEVICE=,ADDRESS=
Defaults:    LINSIZE=80,CHKSUM=16,END=NO
```

Operand	***Description***
DEVICE=	S1S1, Series/1-to-Series/1 Attachment (RPQ D02241 or D02242).
ADDRESS=	The address (in hexadecimal) of the device.
LINSIZE=	The maximum length of an input or output line for the device. The value of this operand can be less than the maximum which the device can accommodate, but the value cannot be increased dynamically. The default is LINSIZE=80.
CHKSUM=	The number of data bytes from which you wish the system to generate a checksum total when the Series/1-to-Series/1 Attachment performs data transfer. Specify 0 if you do not want the system to perform error detection. Specify 2, 4, 8, 16, or 32 if you want error detection. The default is CHKSUM=16. **Note:** If you specify error detection, the data transfer rate is affected.
END=	YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

Series/1-to-Series/1 *(continued)*

Example and Defaults

In the following example, the default assignments for DEVICE=S1S1 are shown as if they were explicitly coded in the TERMINAL statement. If an operand is not shown, then it is not relevant for the device. Address assignment is for illustration only.

Example: Series/1-to-Series/1 TERMINAL statement.

```
label     TERMINAL   DEVICE=S1S1,ADDRESS=25
```

is equivalent to:

```
label     TERMINAL   DEVICE=S1S1,ADDRESS=25,LINSIZE=80,      C
                CHKSUM=16,END=NO
```

TIMER - Define System Timer Features

TIMER defines the #7840 Timer Feature to be used as the system timers in the generated system. There are two timers on the #7840 Timer feature card. One is used for time of day recording and the other is used for interval timing. Only one TIMER statement is required to define both timers. When you list out the devices supported by your operating system using $IOTEST (LS), two timers are shown.

This statement is used only for defining the #7840 timer. If the system has a native timer (4954 and 4956 processors) that is used instead of the #7840 timer feature card, it is not necessary to use this or any other statement. The native timer and the #7840 timer are mutually exclusive.

Syntax:

```
blank          TIMER          ADDRESS=

Required:      ADDRESS=
Defaults:      None
```

Operand	*Description*
ADDRESS=	The hexadecimal address of the #7840 Timer Feature.

Example: Both timers are defined by a single TIMER definition statement.

```
          TIMER     ADDRESS=40
```

Appendix B. Customizing Adapters with Hardware Jumpers

Each hardware feature has hardware jumpers that are used to customize it. You need to become familiar with these jumpers in order to configure the hardware for your system correctly. Before you actually connect terminals and modems, also refer to *IBM Series/1 Communications Feature Description*, GA34-0028. If your terminal configuration includes the IBM 3101 Display Terminal, see both Appendix C, "3101 Configuration Information" on page IS-249 and*IBM 3101 Display Terminal Description*, GA18-2033.

After you have customized the hardware features properly, you must define the terminals to the Event Driven Executive system using the TERMINAL statement; see Appendix A, "System Definition Statements" on page IS-145.

ACCA Adapters

Each ACCA adapter feature has hardware jumpers that you use to customize it. Be sure the hardware is configured correctly prior to defining the software interface.

For information on customizing adapters for use with the 3101 Display Terminal, see Appendix C, "3101 Configuration Information."

ACCA Adapters *(continued)*

Some general rules for hardware jumpers are:

- For direct connect terminals:
  - You usually should jumper Data Terminal Ready (DTR)
  - You usually should jumper Request to Send (RTS)
  - You should jumper Carrier Detect (CD) only when the terminal provides Request to Send (RTS).
- For leased lines using modems:
  - You jumper Data Terminal Ready (DTR) only when Event Driven Executive application programs do not control the modem
  - You jumper Request to Send (RTS) only if the modem provides a steady Clear to Send (CTS) signal
  - You jumper Carrier Detect (CD) only if the modem supports this feature.
- For switched lines using modems:
  - You jumper Data Terminal Ready (DTR) only when Event Driven Executive application programs do not control the modem
  - You jumper Request to Send (RTS) only if the modem provides a steady Clear to Send (CTS) signal
  - You jumper Carrier Detect (CD) only if the modem supports this feature.

You should install speed range jumpers in accordance with instructions in the *Communications Feature Description.*

The Multifunction Attachment (#1310) does not contain hardware jumpers for speed range, DTR, RTS or CD. The system makes required connections internally at IPL time according to the configuration for the TERMINAL statement for the device.

Interprocessor Communications using #1610 Controller

In addition to defining the #1610 controller to the Event Driven Executive with the TERMINAL statement, you should set the hardware jumpers on the attachment according to the *IBM Series/1 Communications Feature Description*, GA34-0028.

For a direct processor interconnection:

- You jumper Data Terminal Ready (DTR)
- You jumper Request To Send (RTS)
- You jumper Low or High speed range depending on the bit rate you choose (100 to 9600 baud).

Be sure to use the right cables for the type of attachments you are interconnecting. For a direct Series/1-to-Series/1 connection, use the local Communication Cable (feature #2056) for one side and the EIA Data Set cable (feature #2057) for the other in order to interchange the Receive/Transmit lines: Data Set Ready (DSR)/Data Terminal Ready (DTR) and Request To Send (RTS)/Clear To Send (CTS). The #2056 cable allows attachment to a modem (male 25-pin type D connector); the #2057 cable allows attachment to a terminal (female 25-pin type D connector).

If only one cable type is available, you must cross the following lines of the 25-pin type D connectors:

Pin number (connector 1)	to	Pin number (connector 2)
1	Protective Ground XMT	1
2	Transmit Data (X or T)	3
3	Receive Data (REC)	2
4	Request to Send (RTS)	5
5	Clear to Send (CTS)	4
6	Data Set Ready (DSR)	20
7	Signal Ground	7
20	Data Terminal Ready (DTR)	6

For a Series/1-to-IBM 5100 connection, you may use the #2056 cable.

Terminals Connected Using Digital I/O #1560

Terminal support is provided for digital I/O devices such as the Tektronix[4] 4010 series of display terminals equipped with the General Purpose Parallel Interface (Tektronix Custom Feature Number CM021-0109-03; with cable: CM012-0541-00) or terminals having equivalent hardware interfaces. The software provides addressing logic enabling up to eight terminals to be shared on one digital input group and one digital output group, with one process interrupt bit for each terminal.

The parallel interface is intended to connect directly to the integrated digital input/output feature (#1560) or the 4982 nonisolated digital input/output features. This interface consists of a driver and a receiver card, each of which has several selectable options. These options allow you to customize the interface to your requirements. You must refer to the manufacturer's manuals for detailed installation procedures.

The following description is intended only to supplement the manufacturer's manuals and to guide you in the use of the Event Driven Executive terminal support on the Series/1. You should select the following options:

Receiver Card	
INTR (interrupt)	PROG
ADDRESS	000(0)-111(7) to match TERMINAL definition
PERM ADD	OFF
PARITY	EVEN
DELAY	3.5-18 (depends on distance)
LOGIC SENSE (3)	Set all to LOW HANDSHAKE CONTROL DATA
THRESHOLD	+2 volts
MASTER OPTION	None

4 Trademark of Tektronix, Inc.

Terminals Connected Using Digital I/O #1560 *(continued)*

Driver Card	
LOGIC SENSE (4)	Set as shown STATUS HIGH HANDSHAKE HIGH INTERRUPT LOW DATA HIGH
INTERRUPT CHANNEL	Use INTR
AUX TSUP	OUT
ECHO	OUT
PARITY	EVEN, BIT 8 IN AB to A, CD to D

Before you can use the terminal with the computer, some other considerations are necessary. As noted above, you must use the common interrupt line (INTR). Select the interrupt line (0 – 7) corresponding to the terminal address. If you attach fewer than eight terminals, you will not use some of the interrupt lines. You must terminate all digital input and process interrupt lines for proper operation. If you use only one terminal, the manufacturer may have installed the DI terminations. With multiple terminals, you should terminate all DI and PI lines at the computer. A 1000-ohm resistor across the DI and PI inputs is recommended. You must set the Baud Rate Selection Switch to the "stand by" position and the J261 Connector Switch to "interface."

If you switch on the terminal and get no response, you will have to reset it. To do so, hold down the SHIFT and RESET keys and switch the LOCAL/LINE switch to LOCAL then to LINE.

Since all Event Driven Executive terminal input/output is done with upper-case ASCII character codes, you should activate the TTY LOCK key when using the terminal with the Series/1.

The last items for special discussion are the GIN mode and the PAGE FULL BREAK strap options on the terminal control card (TC-2). You should set the GIN termination to NONE. Thus, when you use GIN mode, you must press the appropriate key followed by the carriage return (CR). You may set the PAGE FULL BREAK termination to either OUT or IN, depending on your preference. If it is IN, the terminal will always stop when it reaches a full page condition. Then you must press the PAGE RESET key in order to continue. If it is OUT, the terminal will automatically go to the home address and continue printing without erasing the screen.

Customizing Adapters with Hardware Jumpers

Multifunction Attachment (#1310)

The Multifunction Attachment (MFA) is supported by both the EDX terminal support I/O instructions and the EDX BSCAM.

The MFA card has four ports for connecting data communications equipment. You can connect the #1310 adapter to the following devices:

- 3101C or 3101B display terminals
- 4975 printers
- a binary synchronous communications line.

The use of the RS-232 interface with the 3101, 4975, or a BSC line requires that you connect the line to Port 0. Port 0 of the MFA can operate in either RS-232 or RS-422 interface mode. Ports 1, 2, and 3 are limited to operations via the RS-422 interface.

You may substitute the #1310 adapter for other asynchronous adapters subject to the limitations on the interface requirements and supported terminal types stated above.

Appendix C. 3101 Configuration Information

Connecting the 3101 to the Series/1

The IBM 3101 Display Terminal is supported under the Event Driven Executive in both character and block mode. This appendix presents information to aid you in planning the link between the Series/1 and the 3101.

The 3101 is connected to the Series/1 by the attachment features supported by the Event Driven Executive Version 4. For each attachment and type of interface, it is necessary to set the 3101 switches, have the attachment card physically jumpered, connect the cables, and specify the appropriate TERMINAL statement (if applicable).

With the exception of the #7850 Teletypewriter Adapter, you can use the attachment features with the 3101 in either character or block mode as follows:

Feature	Feature number	Character mode	Block mode
Teletypewriter Adapter	7850	Yes	No
Multifunction Attachment	1310	Yes	Yes
Asynchronous Communications Single-Line Controller	1610	Yes	Yes
Asynchronous Communications 8-Line Controller with 4-Line Adapter	2091/ 2092	Yes	Yes
Feature Programmable 8-Line Controller with 4-Line Adapter	2095/ 2096	Yes	Yes

Figure 14. 3101 attachment features

Connecting the 3101 to the Series/1 *(continued)*

For each of these attachment features, there are many 3101 configurations available. Refer to "Configuring Your 3101" on page IS-258 for sample configurations for each attachment card option, the cable connections, the 3101 switch settings, and the TERMINAL statements for each interface/attachment.

3101 Display Terminal in Character Mode

You can connect the IBM 3101 Display Terminal (in character mode) to the Series/1 through five attachment features:

- #7850 teletypewriter adapter
- #1310 adapter
- #1610 controller
- #2091 controller with #2092 adapter
- #2095 controller with #2096 adapter.

In the following discussion, all connections are direct with no intervening modem. For a discussion of leased and switched lines using modems, refer to Appendix B, "Customizing Adapters with Hardware Jumpers" on page IS-243.

Connecting the 3101 to the Series/1 *(continued)*

Attachment via the #7850 Teletypewriter Adapter

Switch settings: For attachment via the #7850 adapter with an EIA interface, you may set the 3101 setup switches as follows:

	Group 1								Group 2							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
on			X				X									
off	X	X		X	X	X		X	X	X	X	X	X	X	X	X

	Group 3								Group 4							
	1	2	3	4	5	6	7	8	1	2	3	4	5	6	7	8
on	X		X	X					X			X	X			X
off		X			X	X	X	X		X	X			X	X	

Attachment options: The input selection jumpers for the #7850 with an EIA interface should be set to 010 indicating EIA with input interpreted as minus=data mark. In the switch settings above, the Group 4 3101 setup switch settings indicate a speed of 9600 bps. You must set the #7850 rate selection jumpers to 1111 to indicate the same speed to the adapter.

The switch settings and attachment options correspond to the configurations numbered 1 and 2 in Figure 19 on page IS-261.

Cable connections: Figure 15 shows the 3101 (character mode) attachment with the #7850 teletypewriter adapter.

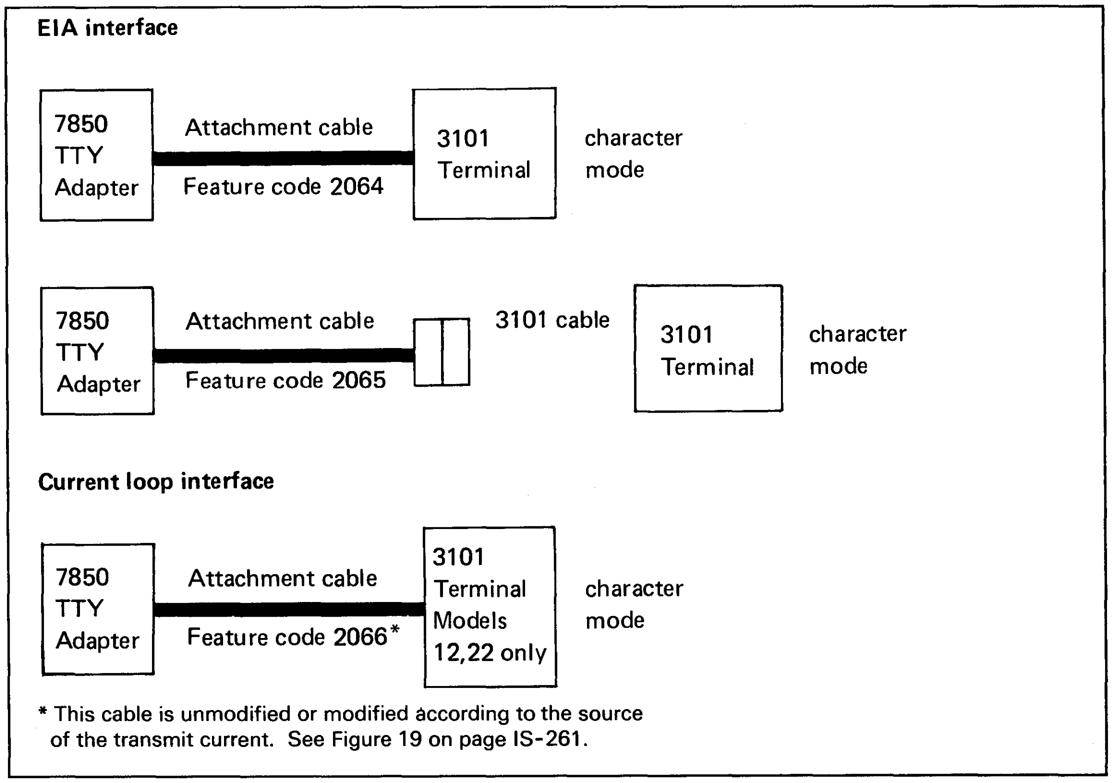


Figure 15. 3101 Terminal with #7850 Adapter

Connecting the 3101 to the Series/1 *(continued)*

Attachment via #1310, #1610, #2091 with #2092, or #2095 with #2096

Switch settings: For attachment via the #1610 controller, the #2091 controller with #2092 adapter, or the #2095 controller with #2096 adapter, you may set the 3101 setup switches as follows:

Group 1	1	2	3	4	5	6	7	8
on		X	X	X			X	
off	X				X	X		X

Group 2	1	2	3	4	5	6	7	8
on								
off	X	X	X	X	X	X	X	X

Group 3	1	2	3	4	5	6	7	8
on	X		X	X				
off		X			X	X	X	X

Group 4	1	2	3	4	5	6	7	8
on	X			X	X			X
off		X	X			X	X	

These switch settings and "Attachment options:" on page IS-253 correspond to the configurations numbered 4, 5, and 6 in Figure 19 on page IS-261.

The switch settings for the #1310 are the same except that switch 3 of group 1 must be in the "off" (RS-422) position if the 3101 terminal is not connected to Port 0. A 3101 attached to Port 0 of a #1310 can operate via either the RS-232 or RS-422A interface. Only 3101 model 23 supports both block mode and the RS-422A interface. Model 13 supports the RS-422A interface but not block mode.

Consider the following points when setting the 3101 setup switches for direct-connect character mode operation:

Group 1, switch 1	Set for character mode operation
Group 1, switch 4	Set for PRTS (permanent request to send)
Group 1, switch 6 & 7	Set for CR (carriage return) as the line turnaround character; the system uses CR as a default line turnaround character for ACCA devices
Group 1, switch 8	Set for mono case. User application programs may use dual case if desired.
Group 2, switch 1	Set for two stop bits for the Event Driven Executive
Group 2, switch 2 & 3	Set for space parity; the system uses space parity for default control characters for ACCA devices.

Connecting the 3101 to the Series/1 *(continued)*

Attachment options: The jumpers for the #1610 controller, the #2091 controller with #2092 adapter, and the #2095 controller with #2096 should have Carrier Detect, Data Terminal Ready, and Request to Send jumpered on. Also, the HIGH- or LOW-speed option must be jumpered to reflect the speed set in the 3101 setup switches. In the switch settings above, the speed is 9600 bps. The RANGE and BITRATE operands on the TERMINAL definition statement must also be compatible with the jumpers and 3101 setup switches.

In addition, for the #2095/#2096, you must set the EIA/TTY jumpers properly to reflect your physical configuration as either an RS-232C (EIA) or current loop (TTY) interface.

The #1310 adapter does not have speed range, DTR RTS, or CD jumpers. However, the BITRATE operand on the TERMINAL configuration statement must be compatible with the 3101 setup switches.

Cable connections: See Figure 17 on page IS-256 when your 3101 is attached through the #1610 or #2091/2092.

Operator Input and Internal Code Considerations

Special considerations that must be given to operator input and internal code representation are summarized in Figure 16 .

		Characters received in the Series/1 with Space Parity set in the 3101 setup switches		
		Device=ACCA		Device=TTY
Operator function	Key on 3101 in character mode	#1610 or #2091 with #2092	#2095 with #2096	#7850
		EBASC	ASCII	ASCII
ATTENTION	PF8	X'D816B0'	X'1B680D'	X'1B680D'
ENTER	↵ (Key above SEND key)	X'B0'	X'0D'	X'0D'
BACKSPACE (character delete)	← (top row, not bottom row)	X'10'	X'08'	X'08'
LINE DELETE	DEL	X'FE'	X'7F'	X'7F'

Figure 16. 3101 Internal Code Representations

The ECHO=NO and PROTECT=YES operands on the READTEXT instruction (for suppression of input text) have no effect when the 3101 is attached via the #1310 adapter, the #1610 controller, the #2091 controller with #2092 adapter, or the #2095 controller with #2096 adapter.

Connecting the 3101 to the Series/1 *(continued)*

Special Considerations for 3101C Use with Half Duplex Modems

While the majority of the 3101 set-up switches remain unchanged when you use half-duplex modems, there are some special requirements for these configurations. The PRTS/CRTS switch must be in the CRTS (controlled request to send) or OFF position. The ESC key will no longer function as the attention key; instead, any PF key may be used as the attention key.

These differences are necessary to allow the 3101 to adhere to the line turnaround protocol observed by half-duplex modems.

3101 Display Terminal in Block Mode

You can connect the IBM 3101 Display Terminal (in block mode) to the Series/1 via four attachments:

- the #1310 adapter
- the #1610 controller
- the #2091 controller with #2092 adapter
- the #2095 controller with #2096 adapter.

Switch Settings

For attachment of the 3101 through these attachment features, you may set the 3101 setup switches as follows:

	Group 1							
	1	2	3	4	5	6	7	8
on	X	X	X	X			X	
off					X	X		X

	Group 2							
	1	2	3	4	5	6	7	8
on				X				
off	X	X	X		X	X	X	X

	Group 3							
	1	2	3	4	5	6	7	8
on	X							
off		X	X	X	X	X	X	X

	Group 4							
	1	2	3	4	5	6	7	8
on	X			X	X			X
off		X	X			X	X	

These switch settings and "Attachment Options" on page IS-255 correspond to the configurations numbered 23, 25, and 26 in Figure 20 on page IS-265.

The switch settings for the #1310 are the same except that switch 3 of group 1 must be in the "off" (RS-422) position if the 3101 terminal is not connected to Port 0. A 3101 model 13 or 23 attached to Port 0 of a #1310 can operate via either the RS-232 or RS-422A interface. Only 3101 model 23 supports both block mode and the RS-422A interface.

Consider the following points when you are setting the 3101 setup switches for block mode operation:

Group 1, switch 1	Set for block mode operation.
Group 1, switch 4	Set for PRTS (permanent request to send) for full duplex modem.
Group 1, switch 6 & 7	Set for CR (carriage return) as the line turnaround character; the system uses CR as a default line turnaround character for ACCA devices.
Group 1, switch 8	Set for mono case. Most IBM system utility programs require the mono case switch setting. User application programs may use dual case if you want.
Group 2, switch 1	Set for two stop bits for the Event Driven Executive.
Group 2, switch 2 & 3	Set for space parity; the system uses space parity for default control characters for ACCA devices.
Group 2, switch 4	Set for Send Line Option so that pressing the SEND key activates the Send Line function. With this switch setting, the system defines the SEND key as the enter key for the 3101 in block mode.
Group 2, switch 6	Set for Null Suppress Off so that input fields are always of a known length.

Attachment Options

The jumpers for the #1610 controller, the #2091 controller with #2092 adapter, and the #2095 controller with #2096 adapter should have Carrier Detect, Data Terminal Ready, and Request to Send jumpered on unless the configuration uses a modem. Also, you must jumper the HIGH- or LOW-speed option to reflect the speed set in the 3101 setup switches. In the switch settings above, the speed is 9600 bps. The RANGE and BITRATE operands on the TERMINAL definition statement must also be compatible with the jumpers and 3101 setup switches.

In addition, for the #2095/2096, you should jumper EIA/TTY for either an EIA interface or a current loop (TTY) interface.

The #1310 does not have speed range, DTR, RTS, or CD jumpers. However, the BITRATE operand of the TERMINAL definition statement must be compatible with the 3101 setup switches.

Connecting the 3101 to the Series/1 *(continued)*

Cable Connections

Figure 17 shows the 3101 (character or block mode) attachment with the #1610 controller or the #2091 controller and the #2092 adapter. Figure 18 on page IS-257 shows the 3101 (character or block mode) attachment with the #1310, #2095 controller with the #2096 adapter (EIA interface), and the #2095 with RPQ D02350.

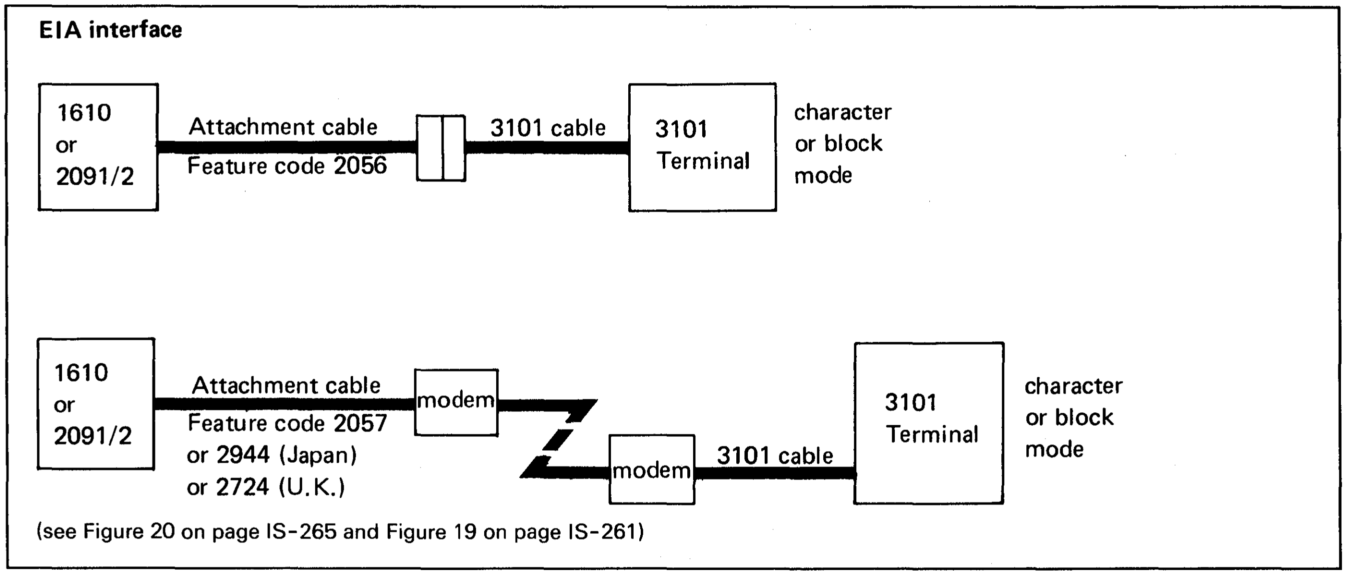
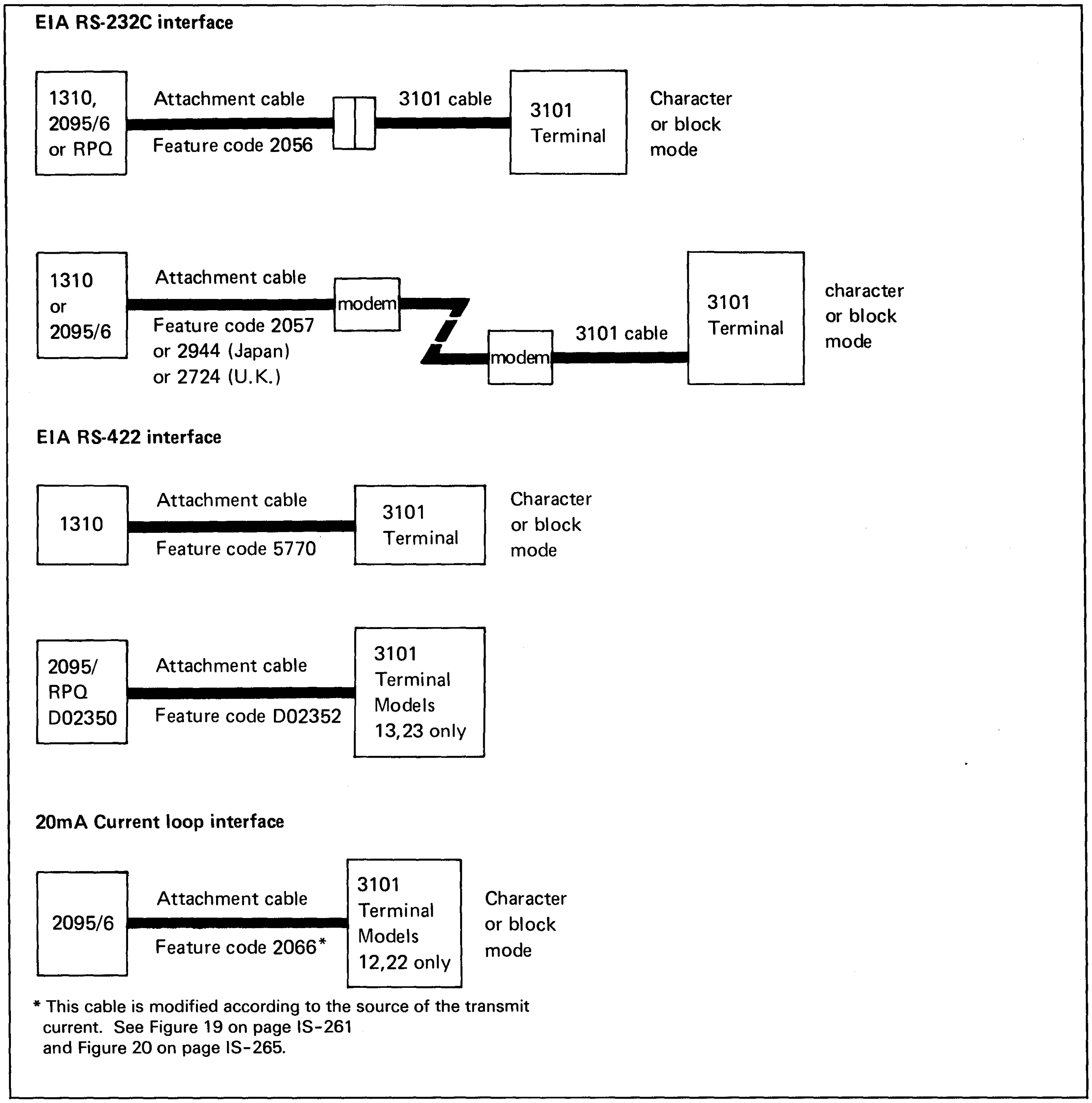


Figure 17. 3101 Terminal with #1610 or #2091/#2092

Connecting the 3101 to the Series/1 *(continued)*



* This cable is modified according to the source of the transmit current. See Figure 19 on page IS-261 and Figure 20 on page IS-265.

Figure 18. 3101 Terminal with #1310, #2095/#2096 and #2095/RPQ D02350

Connecting the 3101 to the Series/1 *(continued)*

Operator Input and Internal Code Considerations

For the 3101 Display Terminal in block mode, the system defines the SEND key as the enter key (with the Send Line Option setup switch set so that the SEND key acts as the Send Line key). In addition, the system default for the Attention key is the PF8 key.

When the 3101 is connected via the #2095/#2096 attachment, pressing the PF8 key places the value X'1B68' into Series/1 storage (with space parity set, and with CR set as the line turnaround character in the setup switches).

When the 3101 is connected via the #1610 or #2091/#2092 attachment, pressing the PF8 key places the value X'D816' into Series/1 storage (with space parity set, and with CR set as the line turnaround character in the setup switches).

Configuring Your 3101

To aid you in configuring your 3101, two figures are presented. Figure 19 on page IS-261 shows 22 configurations for the 3101 in character mode; Figure 20 on page IS-265 shows 17 configurations for the 3101 in block mode.

The individual configurations are numbered sequentially starting with the first configuration in Figure 19 and concluding with the configurations in Figure 20. The configuration numbers appear in the left margin on the even numbered page and in the right margin on the odd numbered page of the charts. These numbers can be used when referring to a particular configuration.

Select the appropriate figure based upon the transmission mode (character or block) of your 3101. Locate the appropriate entry in the figure based upon the type of interface and the Series/1 attachment feature. Now you can read across the figure to find the:

- Series/1 attachment feature used for a specific interface
- Attachment options of the attachment feature
- Feature code and part number of the Series/1 cable
- Part number of the 3101 cable
- 3101 models supported by a specific configuration
- 3101 switch setting for a specific configuration
- TERMINAL definition statements specified at system generation corresponding to a specific configuration.

Configuring Your 3101 *(continued)*

Determine the type of interface and the attachment feature installed in your system. Communicate the attachment options required to your IBM Customer Engineer during Series/1 or attachment feature installation. He will install the appropriate options on the attachment feature card. You specify the appropriate TERMINAL definition statement and set the 3101 setup switches according to the type of configuration.

While the configurations shown in Figure 19 and Figure 20 are typical, they were chosen for illustrative purposes only. There may be other 3101 configurations depending on the attachment feature and type of interface appropriate for your particular application.

Be sure to consider the numbered notes indicated in the column headings of each chart. These numbers correspond to the list of notes immediately following the charts.

This page intentionally left blank.

	Type of interface		Series/1 attachment features	Attachment options (jumpers)								Attachment cable	
				Typical device address	Range (bitrate for 7850)	Request to send	Data terminal ready	Carrier detect	TTY-EIA	Input select	Isolated non-isolated	Feature code	Part number or equivalent
				Notes 1-3	Notes 2-3	Note 3						Note 4	
(1)	Local	EIA RS232C	7850 TTY	00	9600 (Jumper= 1111)	–	–	–	–	EIA-Minus =Data Mark (Jumper= 010)	Not used	2064	1632924
(2)			7850 TTY	00	9600 (Jumper= 1111)	–	–	–	–	EIA-Minus =Data Mark (Jumper= 010)	Not used	2065	4411751
(3)			1310 MFA	58	–	–	–	–	–	–		2056	1632211
(4)			1610 Single line control ACCA	08	High (Jumper= Medium)	On	On	On	–	–	–	2056	1632211
(5)			2091/2092 Multi-line ACCA	60	High	On	On	On	–	–	–	2056	1632211
(6)			2095/2096 Feature programmable multi-line	68	High	On	On	–	EIA	–	–	2056	1632211
(7)		EIA 422	1310 MFA	58	–	–	–	–	–	–	–	5770	6844552
(8)			2095 with RPQ D02350	68	High	On	On	On	–	–	–	D02352	6844552
(9)		Current loop (3101 supplied current)	7850 TTY	00	9600 (Jumper= 1111)	–	–	–	–	Contact sense closed =data mark (Jumper= 000)	Isolated no jumper (Jumper= 0)	2066 Un-modified	6839455 Pins: A07-17 A03-18,B05-07 A01-25, 15-23 24-1
(10)			2095/2096 Feature programmable multi-line	68	High	On	On	On	TTY	–	–	2066 Modified	6839455 Pins: A04-17,A01-B01 B05-18,A03-B03 B04-25,A05-07 15-23, 1-24
(11)		Current loop (attachment supplied current)	7850 TTY	00	9600 (Jumper= 1111)	–	–	–	–	Contact sense closed =data mark (Jumper= 000)	Non-ilosated (Jumper= 1)	2066 Modified	6839455 Pins: A02-15,A03-24 B01-25,B05-17
(12)			2095/2096 Feature programmable multi-line	68	High	On	On	On	TTY	–	–	2066 Modified	6839455 Pins: B06-25,A07-A04 B05-24,B07-B04 A06-17,A01-B01 A05-15,A03-B03
(13)		Current loop (3101 and attachment both supply current)	7850 TTY	00	9600 (Jumper= 1111)	–	–	–	–	Contact sense closed =data mark (Jumper =000)	Isolated no jumper (Jumper =0)	2066 Modified	6839455 Pins: A01-25,A02-15 A03-18,B05-17 1-24
(14)			2095/2096 Feature programmable multi-line	68	High	On	On	On	TTY	–	–	2066 Modified	6839455 Pins: A06-17,A04-A07 A05-15,A01-B01 B05-18,A03-B03 B04-25, 1-24

Figure 19 (Part 1 of 4). Configuration Matrix for 3101 in Character Mode

Configuring Your 3101 *(continued)*

Direct attach or modem	3101 Cable part number	3101 Model supported	3101 Switch settings corresponding to type of configuration (1 = On, 0 = Off)				Terminal configuration statements corresponding to the type of configuration	
			Group 1	Group 2	Group 3	Group 4		
	Note 5	Note 6	Note 7				Note 8	
Direct	–	10,12,13 20,22,23	0010 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=TTY,ADDRESS=00,MODE=3101C	(1)
Direct	5640736	10,12,13 20,22,23	0010 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=TTY,ADDRESS=00,MODE=3101C	(2)
Direct	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=58,ADAPTER=MFA, C BITRATE=9600,LMODE=LOCAL,MODE=3101C	(3)
Direct	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=08,MODE=3101C, C BITRATE=9600,RANGE=HIGH	(4)
Direct	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=60,MODE=3101C, C BITRATE=2400,ADAPTER=FOUR, C RANGE=HIGH	(5)
Direct	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101C, C CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, C LF=OA,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH	(6)
Direct	–	13,23	0101 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=58,ADAPTER=MFA, C BITRATE=9600,LMODE-RS422,MODE=3101C	(7)
Direct	–	13,23	0101 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,ADAPTER=EIGHT, C CODTYPE=ASCII,BITRATE=9600, C LMODE=LOCAL,MODE=3101C	(8)
Direct	–	12,22	0000 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=TTY,ADDRESS=00,MODE=3101C	(9)
Direct	–	12,22	0101 0010	0000 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101C, C CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH	(10)
Direct	–	12,22	0000 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=TTY,ADDRESS=00,MODE=3101C	(11)
Direct	–	12,22	0101 0010	0000 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101C, C CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH	(12)
Direct	–	12,22	0000 0010	0000 0000	1011 0000	1001 1001	TERMINAL DEVICE=TTY,ADDRESS=00,MODE=3101C	(13)
Direct	–	12,22	0101 0010	0000 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101C, C CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH	(14)

Figure 19 (Part 2 of 4). Configuration Matrix for 3101 in Character Mode

	Type of interface		Series/1 attachment features	Attachment options (jumpers)								Attachment cable	
				Typical device address	Range (bitrate for 7850)	Request to send	Data terminal ready	Carrier detect	TTY-EIA	Input select	Isolated non-isolated	Feature code	Part Number or equivalent
				Notes 1-3	Notes 2-3	Note 3						Note 4	
(15)	**Remote**	**EIA RS232C (Full duplex modem non-switched)**	1610 Single line control ACCA	08	High (Jumper= medium)	On	On	On	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(16)			2091/2092 Multi-line ACCA	60	High	On	On	On	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(17)			2095/2096 Feature programmable multi-line	68	High	On	On	On	EIA	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(18)		Half-duplex only	1310 MFA	58	–	–	–	–	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(19)		**EIA RS232C (Full duplex modem switched)**	1610 Single line control ACCA	08	High (Jumper= medium)	Off	Off	Off	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(20)			2091/2092 Multi-line ACCA	60	High	Off	Off	Off	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(21)			2095/2096 Feature programmable multi-line	68	High	Off	Off	Off	EIA	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(22)		Half-duplex only	1310 MFA	58	–	–	–	–	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744

Figure 19 (Part 3 of 4). Configuration Matrix for 3101 in Character Mode

Configuring Your 3101 *(continued)*

Direct attach or modem	3101 Cable part number	3101 Model supported	3101 Switch settings corresponding to type of configuration (1 = On, 0 = Off)				Terminal configuration statements corresponding to the type of configuration	
			Group 1	Group 2	Group 3	Group 4		
	Note 5	Note 6	Note 7				Note 8	
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=08,MODE=3101C, C BITRATE=1200,RANGE=HIGH	(15)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=60,MODE=3101C, C BITRATE=1200,ADAPTER=FOUR, C RANGE=HIGH	(16)
Modem	5640736	10,12,13, 20,22,23,	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101C C CODTYPE=ASCII,BITRATE=1200,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, RANGE=HIGH	(17)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=58, C ADAPTER=MFA,MODE=3101C	(18)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=08,MODE=3101C, C BIRATE=1200,LMODE=SWITCHED, C RANGE=HIGH	(19)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=60,MODE=3101C, C BITRATE=1200,LMODE=SWITCHED, C ADAPTER=FOUR,RANGE=HIGH	(20)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101C, C CODTYPE=ASCII,BITRATE=1200,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH,LMODE=SWITCHED	(21)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=58,MODE=3101C, C LMODE=SWITCHED,ADAPTER=MFA	(22)

Figure 19 (Part 4 of 4). Configuration Matrix for 3101 in Character Mode

	Type of interface		Series/1 attachment features	Attachment options (jumpers)								Attachment cable	
				Typical device address	Range (bitrate for 7850)	Request to send	Data terminal ready	Data carrier detect	TTY-EIA	Input select	Isolated non-isolated	Feature code	Part Number or equivalent
				Notes 1-3	Notes 2-3	Note 3						Note 4	
(23)	Local	EIA RS232C	1610 Single line control ACCA	08	High (Jumper= medium)	On	On	On	–	–	–	2056	1632211
(24)			1310 MFA	58	–	–	–	–	–	–	–	2056	1632211
(25)			2091/2092 Multi-line ACCA	60	High	On	On	On	–	–	–	2056	1632211
(26)			2095/2096 Feature programmable multi-line	68	High	On	On	On	EIA	–	–	2056	1632211
(27)		EIA RS422	1310 MFA	58	–	–	–	–	–	–	–	5770	6844552
(28)			2095 with RPQ D02350	68	High	On	On	On	EIA	–	–	D02352	6844552
(29)		Current loop (3101 supplied current)	2095/2096 Feature programmable multi-line	68	High	On	On	On	TTY	–	–	2066 Modified	6839455 Pins: A04-17,A01-B01 B05-18,A03-B03 A05-7, 15-23 B04-25, 1-24
(30)		Current loop (attachment supplied current)	2095/2096 Feature programmable multi-line	68	High	On	On	On	TTY	–	–	2066 Modified	6839455 Pins: B06-25,A07-A04 B05-24,B07-B04 A06-17,A01-B01 A05-15,A03-B03
(31)		Current loop (3101 and attachment both supply current)	2095/2096 Feature programmable multi-line	68	High	On	On	On	TTY	–	–	2066 Modified	6839455 Pins: A06-17,A04-A07 A05-15,A01-B01 B05-18,A03-B03 B04-25, 1-24

Figure 20 (Part 1 of 4). Configuration Matrix for 3101 in Block Mode

Configuring Your 3101 *(continued)*

Direct attach or modem	3101 Cable part number	3101 Model supported	3101 Switch settings corresponding to type of configuration (1 = On, 0 = Off)				Terminal configuration statements corresponding to the type of configuration	
			Group 1	Group 2	Group 3	Group 4		
	Note 5	Note 6	Note 7				Note 8	
Direct	5640736	20,22,23	1111 0010	0001 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=08,MODE=3101B, C BITRATE=9600,RANGE=HIGH	(23)
Direct	5640736	20,22,23	1111 0010	0001 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=58,MODE=3101B, C BITRATE=9600,ADAPTER=MFA,LMODE=LOCAL	(24)
Direct	5640736	20,22,23	1111 0010	0001 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=60,MODE=3101B, C BITRATE=2400,ADAPTER=FOUR, C RANGE=HIGH	(25)
Direct	5640736	20,22,23	1111 0010	0001 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101B, C CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH	(26)
Direct	–	13,23	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=58,MODE=3101B, C BITRATE=9600,ADAPTER=MFA, C LMODE=RS422	(27)
Direct	–	13,23	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101B, C BITRATE=9600,ADAPTER=EIGHT, C LMODE=LOCAL,CODTYPE=ASCII	(28)
Direct	–	22	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101B, C CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH	(29)
Direct	–	22	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101B, C CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH	(30)
Direct	–	22	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101B, C CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH	(31)

Figure 20 (Part 2 of 4). Configuration Matrix for 3101 in Block Mode

Configuring Your 3101 *(continued)*

	Type of interface		Series/1 attachment features	Attachment options (jumpers)								Attachment cable	
				Typical device address	Range (bitrate for 7850)	Request to send	Data terminal ready	Data carrier detect	TTY-EIA	Input select	Isolated non-isolated	Feature code	Part Number or equivalent
				Notes 1-3	Notes 2-3	Note 3						Note 4	
(32)	**Remote**	**EIA RS232C (Full duplex modem non-switched)**	1610 Single line control	08	High (Jumper= medium)	On	On	On	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(33)		Half-duplex only	1310 MFA	58	–	–	–	–	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(34)			2091/2092 Multi-line ACCA	60	High	On	On	On	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(35)			2095/2096 Feature programmable multi-line	68	High	On	On	On	EIA	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(36)		**EIA RS232C (Full duplex modem switched)**	1610 Single line control	08	High (Jumper= medium)	Off	Off	Off	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(37)		Half-duplex only	1310 MFA	58	–	–	–	–	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(38)			2091/2092 Multi-line ACCA	60	High	Off	Off	Off	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(39)			2095/2096 Feature programmable multi-line	68	High	Off	Off	Off	EIA	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744

Figure 20 (Part 3 of 4). Configuration Matrix for 3101 in Block Mode

3101 Configuration Information

Configuring Your 3101 *(continued)*

Direct attach or modem	3101 Cable part number	3101 Model supported	3101 Switch settings corresponding to type of configuration (1 = On, 0 = Off)				Terminal configuration statements corresponding to the type of configuration	
			Group 1	Group 2	Group 3	Group 4		
	Note 5	Note 6	Note 7				Note 8	
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=08,MODE=3101B, C BITRATE=1200,RANGE=HIGH	(32)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=58,MODE=3101B, C ADAPTER=MFA	(33)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=60,MODE=3101B, C BITRATE=1200,ADAPTER=FOUR, C RANGE=HIGH	(34)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101B, C CODTYPE=ASCII,BITRATE=1200,ATTN=1B68, C LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, C RANGE=HIGH	(35)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=08,MODE=3101B, C BITRATE=1200,LMODE=SWITCHED, C RANGE=HIGH	(36)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=58,MODE=3101B, C LMODE=SWITCHED,ADAPTER=MFA	(37)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=60,MODE=3101B, C BITRATE=1200,LMODE=SWITCHED, C ADAPTER=FOUR,RANGE=HIGH	(38)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL DEVICE=ACCA,ADDRESS=68,MODE=3101B, C CODTYPE=ASCII,BITRATE=1200, C LMODE=SWITCHED,ATTN-1B68,LF=0A, C CR=0D,PF1-1B61,ADAPTER=FOUR, C RANGE=HIGH	(39)

Figure 20 (Part 4 of 4). Configuration Matrix for 3101 in Block Mode

Configuration Matrix Notes

1. The device addresses shown in Figure 19 on page IS-261 and Figure 20 on page IS-265 are typical. For additional information on device addresses, see the *IBM Series/1 Configurator*, GA34-0042.

   Note: The attachment feature device address jumpered on the attachment card must agree with the ADDRESS= parameter of the TERMINAL definition statement specified during system generation. For a description of the TERMINAL definition statement, see Appendix A, "System Definition Statements" on page IS-145.

2. Speed range adapter jumpers differ between attachment features. The following figure shows the speed range options that must be jumpered for a specific attachment feature. In the figure, N/A means no medium-speed jumper on the #2092 and #2096; no high-speed jumper on the #1610; no speed jumpers on the #1310.

Feature	Speed range (bits/second)	Low speed jumper	Medium speed jumper	High speed jumper
#7850 Teletypewriter Adapter	Bitrate of 110-9600	-	-	-
#1610 Asynchronous Communications Single-Line Controller	110-1200 300-9600	Yes -	- Yes	N/A N/A
#2092 Asynchronous Communications Four-Line Adapter	110-1200 300-2400	Yes -	N/A N/A	- Yes
#2096 Feature Programmable Four-Line Adapter	110-1200 300-9600	Yes -	N/A N/A	- Yes
#1310 Multifunction Attachment	1200-9600	N/A	N/A	N/A

Figure 21. Speed range jumper options

3. The following information refers to the attachment options:

   a. The #7850 TTY attachment feature card does not require speed range selection. However, a specific rate selection (BITRATE) between 110 and 9600 bits per second (bps) must be selected by the use of jumpers on this card.

   b. In all cases where speed range is selectable, the adapter range jumpered must correspond to the RANGE= parameter specified in the TERMINAL definition statement. Note, however, that the #1310 does not have physical speed (BITRATE) jumpers. Refer to the *Machine Logic Diagrams* for specific adapter requirements.

Configuration Matrix Notes *(continued)*

Total system throughput is dependent on the system configuration as follows:

a. The #1610 Asynchronous Communications Single-Line Controller feature can control one line per feature with speeds ranging from 110 to 9600 bps.

b. The #2091 Asynchronous Communications 8-Line Controller feature controls one or two 2092 Asynchronous Communications 4-Line Adapter feature (up to four lines per feature). Speeds from 110 to 2400 bps are supported by the Event Driven Executive with a maximum of 2400 bps on all eight lines.

c. The #2095 Feature-Programmable 8-Line Communications Controller feature supports up to two #2096 Feature-Programmable 4-Line Communication Adapter features (up to four lines per feature). Speeds from 110 to 9600 bps are supported by the Event Driven Executive. However, the maximum aggregate throughput you can have is 64000 bps at 12 bits per character.

For additional information on speed settings for the above mentioned attachment features, see the *IBM Series/1 Customer Site Preparation Manual*, GA34-0050.

4. For additional information regarding adapter jumper requirements, see the *Machine Logic Diagrams* or the *IBM Series/1 Communication Features Theory Diagrams*, SY34-0059.

5. The cable part numbers referenced in Figures 25 and 26 are engineering-change sensitive and are subject to revision. Refer to the *IBM Series/1 Customer Site Preparation Manual*, GA34-0050, for additional cabling information.

6. Cable bill of material (B/M) 5640736 can be ordered with the 3101 Models 10 (AAS110) and 20 (AAS210). For Models 12, 13, 22, and 23, you must order this cable separately.

7. For additional information regarding attachments and configuration requirements of the 3101, see the *Event Driven Executive Language Programming Guide*.

8. The 3101 Group 4 switch settings will vary with different communication line speeds (baud rates). The following table lists the Group 4 switch settings for the baud rates supported by the 3101.

Baud rate	Group 4
100	0000 0000
150	0001 0001
200	0010 0010
300	0011 0011
600	0100 0100
1200	0101 0101
1800	0110 0110
2400	0111 0111
4800	1000 1000
9600	1001 1001

Figure 22. 3101 Supported Baud Rates and Group 4 Switch Settings

For additional information regarding the 3101 setup switches and interface support provided for the 3101, see the *IBM 3101 Display Terminal Description*, GA18-2033.

9. For a description of the TERMINAL definition statement, see the Appendix A, "System Definition Statements" on page IS-145.

   a. For illustrative purposes, the RANGE= parameter in the 3101 configuration matrices is coded as HIGH to emphasize that the RANGE= parameter must equal the SPEED RANGE option jumper on the attachment feature. When specifying your TERMINAL definition statement, you need not code RANGE=HIGH as it is the default.

   b. For DEVICE=ACCA, you can modify the timer values at sysgen time by using the TIMERS parameter of the TERMINAL statement and a list of 4 hexadecimal numbers set to the value desired.

   c. If you have specified LMODE=SWITCHED on the TERMINAL configuration statement for your 3101 Display Terminal, or specified (or defaulted) LMODE=PTTOPT for a leased line connection with modems, you may need to modify the pretransmit and posttransmit timers in the DCB that is contained in the CCB for this terminal.

   For LMODE=SWITCHED, the generated value of the timer 2 transmit word is X'012C'; for LMODE=PTTOPT, the generated value of the timer 2 transmit word is X'0002'. You can obtain the address of the CCB from the link-edit map of your generated system. You can locate the timer 2 transmit word by subtracting X'005E' from the CCB address.

Configuration Matrix Notes *(continued)*

Depending on the modem and line speed (bit rate) selected, you may have to patch the timer 2 transmit word to a higher value. If you do not know the proper value for your modem and line speed, try a high value and then successively lower values for the timer 2 transmit word.

Following is an example of patching the timer 2 transmit word of the DCB in the CCB of an ACCA terminal. In this example, the entry point address of the CCB of the terminal named ACCA08 is X'18E6' as shown in the following sample link edit map.

```
     .
     .
ENTRY    $SYSLOG      0F06
ENTRY    $SYSLOGA     10E0
ENTRY    $SYSPRTR     12C4
ENTRY    MPRINTER     14DC
ENTRY    TTY00        16D4
ENTRY    ACCA08       18E6
ENTRY    ACCA18       1BB8
ENTRY    MLTA60       1DB8
ENTRY    MLTA61       1FCA
     .
     .
```

The timer 2 transmit word is located at address X'18E6' minus X'005E' or X'1888' in the generated system.

The following is a sample of the assembled object code containing the timer 2 transmit word to be altered.

```
  .
  .
  .
1872    2004 0000 0000 0000 0000
187C    0000 0002 1460 0002 0002
1886    000A 012C 0000 0000 0000
1890    0000 001D 1FFE B0B0 B0B0
  .
  .
  .
```

Configuration Matrix Notes *(continued)*

Use $DISKUT2 to patch the timer 2 transmit word from X'012C' to X'0200' as follows:

```
> $L $DISKUT2
$DISKUT2      46P,00:00:00, LP= 8F00

USING VOLUME GD3101

COMMAND (?):  PA
PGM OR DS NAME:  $EDXNUC
$EDXNUC IS A PROGRAM OF HEX SIZE 00003B7E
ADDRESS:  1888
HOW MANY WORDS?  1
(D)EC, (E)BCDIC OR (H)EX?  H

NOW IS:
 1888     012C              |.....)      |

ENTER DATA:  0200

NEW DATA:
 1888     0200              |.....¬      |

OK?  Y
PATCH COMPLETE
ANOTHER PATCH?  N

COMMAND (?):  EN
$DISKUT2 ENDED
```

After completing the previous procedure, the timer 2 transmit word is as follows:

```
   .
   .
   .
1872    2004 0000 0000 0000 0000
187C    0000 0002 1460 0002 0002
1886    000A 0200 0000 0000 0000
1890    0000 001D 1FFE B0B0 B0B0
   .
   .
   .
```

For additional information about the DCB and the timer 2 transmit word, see "Asynchronous Communications" in the *IBM Series/1 Communications Features Description*, GA34-0028.

Notes

Appendix D. Supervisor Module Names (CSECTS)

This appendix contains the names of all the object modules you can include in your supervisor. The first column lists the names of entry points into specific object modules. The second column lists the names of the object module to which the entry point is associated. Other object modules within your supervisor can refer to each entry point.

After you generate your supervisor, check the $XPSLINK link map to see if you have any unresolved EXTRNs. An unresolved EXTRN is caused by a supervisor object module referring to an entry point within another supervisor object module that is not included in your supervisor. Locate the entry point that corresponds to the unresolved EXTRN. The associated module name indicates whether or not you must modify the EDXDEFS or LINKCNTL data sets. A module name of $EDXDEF signifies that you must correct an existing definition statement or that you failed to code a required definition statement in the EDXDEFS data set. All other module names signify which supervisor object module must be included in the LINKCNTL data set. Modify the appropriate data set and run the $JOBUTIL procedure again to regenerate your supervisor.

Note: Following each occurrence of $EDXDEF is the name of the definition statement that is either in error or missing.

Supervisor Module Names (CSECTS)

Entry Point	Object Module
$A	RLOADER
$ACTIVE	RLOADER
$ADPINIT	INITMODS
$ADPOPEN	INITMODS
$AUTOINT	INITMODS
$BITS	EDXALU
$BONOFF	EDXALU
$BSCADDR	EDXSYS
$BSCADS	XPSPART
$BSCAINT	INITMODS
$BSCFDDB	$EDXDEF (BSCLINE)
$BUFF	$EDXTIO
$CANCEL	RLOADER
$CFATTN	$EDXTIO
$CFINIT1	INITMODS
$CFINIT2	INITMODS
$CFPARM	EDXSYS
$CN	RLOADER
$COMPADR	EDXALU
$COMPCNT	EDXALU
$COMPE	EDXALU
$COMPNE	EDXALU
$CP	EDXSVCX
$CVTADS	XPSPART
$DDBC	$EDXDEF
$DSKADS	XPSPART
$DISKINT	INITMODS
$DMDDB	EDXSYS
$DMVOL	EDXSYS
$DPEND	$EDXTIO
$DSKINT1	INITMODS
$DSKINT2	INITMODS
$EDXINIT	EDXINIT1
$EDXTIO	NOTIO
$EDXPTCH	$EDXDEF (SYSTEM)
$EXEC	EDXALU
$EXIODDB	$EDXDEF (EXIO)
$EXIOINT	INITMODS
$FINDE	EDXALU
$FINDNE	EDXALU
$FLTADS	XPSPART
$FRSTACB	EDXSYS
$HOSTINT	INITMODS
$IAMQCB	EDXSYS
$INITMOD	$EDXDEF (SYSTEM)
$INITPRT	$EDXDEF (SYSTEM)
$IOUS	$EDXDEF (SYSTEM)
$IPLHOOK	EDXSYS
$IPLVOL	EDXSYS
$LOADINT	INITMODS
$LOADIN2	INITMODS
$LOGIA	SYSLOG
$LOGPARM	EDXSYS
$LOGTSK	SYSLOG
$MAPAREA	$EDXDEF (SYSTEM)
$MEMSIZE	$EDXDEF (SYSTEM)
$MFAINIT	INITMODS

Entry Point	Object Module
$MFAOPEN	INITMODS
$MSGBUF	FULLMSG
$MSGBUF	MINMSG
$NUMPART	$EDXDEF (SYSTEM)
$PARTSZE	$EDXDEF (SYSTEM)
$PGMCINT	INITMODS
$PHYSTG	$EDXDEF (SYSTEM)
$PRINT2	$EDXTIO
$QIOADS	XPSPART
$SBIOADS	XPSPART
$SBIOINT	INITMODS
$SBPITAB	EDXSYS
$SEGINIT	EDXINIT1
$SFLIST	EDXSYS
$SLOGPRM	SYSLOG
$SLOGTSK	SYSLOG
$SRMADS	XPSPART
$SRTBL	NOSRMGR
$START	EDXSYS
$STORAGE	$EDXDEF (SYSTEM)
$STORINT	INITMODS
$SVCBCTL	EDXSYS
$SVCIBUF	$EDXDEF (SYSTEM)
$SVCLSB	EDXSVCX
$SVCSIA	EDXSVCX
$SVCSTOP	EDXSVCX
$SYSCOM	$EDXDEF (SYSTEM)
$SYSLOG	$EDXDEF (TERMINAL)
$SYSLOGA	$EDXDEF (TERMINAL)
$SYSLOGB	$EDXDEF (TERMINAL)
$SYSLOG1	$EDXDEF (TERMINAL)
$SYSLOG2	$EDXDEF (TERMINAL)
$SYSPRTR	$EDXDEF (TERMINAL)
$TAPEINT	INITMODS
$TESTADR	EDXSYS
$TESTCOM	$DBUGNUC
$TIMEINT	INITMODS
$TIMRTBL	EDXSYS
$TIOADS	XPSPART
$TPDDB	EDXSYS
$TPDDB1	TPCOM
$TPDVADR	$EDXDEF
$TPIA	TPCOM
$TRCLSB	EDXSTART
$TRCSIA	$DBUGNUC
$TRMINIT	INITMODS
$TRMOPEN	INITMODS
$USRATT1	$EDXTIO
$USRATT2	$EDXTIO
$USRATT3	$EDXTIO
$XPSBCTL	EDXSYS
$XPSHEAD	EDXSYS
$XPSINIT	EDXINIT1
$XPSTABL	EDXSYS
$XTSK	X21CMD
$X21IA	$X21IA
$4013INT	INITMODS

$4978INT	INITMODS
$4980INT	INITMODS
$4980OPN	INITMODS
##ZERO##	EDXSYS
##ZERO##	XPSSYS
#ACLOSE	D4969A
#ALLSCB	NOSRMGR
#ATTN10	D49624
#ATTN10A	D4963A
#ATTN35	D49624
#ATTN40	D49624
#ATTN50	D4966A
#ATTN51	D4966A
#ATTN55A	D4966A
#ATT1024	$D1024
#BSCENTR	BSCAM
#BSCIA	BSCAM
#BSCIORT	BSCAM
#BSCSTRT	BSCAM
#CMDSETU	EDXALU
#CMDSTUP	EDXALU
#CNTLEND	D4963A
#CTLXFER	EDXTIO
#CURCTL	EDXTIO
#DELRT	DISKIO
#DELSCB	NOSRMGR
#DEQBSC	BSCAM
#DEQLBNK	NOLRDCHK
#DEQT	EDXTERMQ
#DISKCHN	DISKIO
#DISKRW	DISKIO
#DISKTBL	DISKTBL
#DQTERMS	EDXTERMQ
#DQTERMS	NOTIO
#DQTRMIN	EDXTERMQ
#DQTRMIN	NOTIO
#DSKCAC	D4963A
#DSKER13	DISKIO
#DSKIO00	DISKIO
#DSKIO99	$DISKIO
#DSKPOST	$DISKIO
#EBBICVT	EDXTIO
#ENDATTN	EDXTIO
#ENQLBNK	NOLRDCHK
#ENQT	EDXTERMQ
#FLIHPST	$D4969A
#FLIH04	$DISKIO
#FLIH36A	D49624
#FLIH40	D4963A
#FLIH50	D49624
#FLIH54	D49624
#GETVAL2	EDXTIO
#GETVAL4	EDXTIO
#IAVIRT	IOSVIRT
#IFB	EDXALU
#IFDW	EDXALU
#IFTEST	EDXALU
#IFTEST1	EDXALU
#IFW	EDXALU
#IOGPIB	IOSGPIB
#IOSPEND	IOSPOOL
#IOSPWR	IOSPOOL
#IOSTERM	IOSTERM
#IOVIRT	IOSVIRT
#IO4973	IOS4974
#IO4974	IOS4974
#IO4975	IOS4974
#IO4978	IOS4979
#IO4979	IOS4979
#IO5219	IOS4974
#IO5224	IOS4974
#IO5225	IOS4974
#KBTASK	EDXTIO
#MESSAGE	FULLMSG
#MESSAGE	MINMSG
#MSGBUF	FULLMSG
#NOP	EDXALU
#NXTCMDA	$EDXTIO
#PRSKSP	EDXTIO
#PRTEXT	EDXTIO
#PRTNUM2	EDXTIO
#PRTNUM4	EDXTIO
#PWRAM80	PWRM80
#QUESTON	EDXTIO
#QUEUEIO	#QUEUEIO
#QUTRMIN	EDXTERMQ
#RDGPIB	IOSGPIB
#RDTEXT	EDXTIO
#RDTEXTL	EDXTIO
#SAI	SBAI
#SAIA	SBAI
#SAIS	SBAI
#SAIX	SBAI
#SAO	SBAO
#SAOA	SBAO
#SAOX	SBAO
#SBCOM	SBCOM
#SBPI	SBPI
#SDI	SBDIDO
#SDIA	SBDIDO
#SDIS	SBDIDO
#SDIX	SBDIDO
#SDO	SBDIDO
#SDOA	SBDIDO
#SDOP	SBDIDO
#SDOS	SBDIDO
#SDOX	SBDIDO
#SLOGTSK	SYSLOG
#STATS78	IOS4979
#TAPEEND	$D4969A
#TERMINT	EDXTIO
#TERMINT	NOTIO
#TERMOUT	EDXTIO
#TERMOUT	NOTIO
#TPEIO00	D4969A
#TPEIO99	$D4969A
#WRGPIB	IOSGPIB
#4966B	D4966A
#624ATTN	D49624
#624DCB	D49624

Supervisor Module Names (CSECTS)

#624ERP	D49624
#624INIT	D49624
#63ADCB	D4963A
#63ATTN	D4963A
#63ERP	D4963A
#63INIT	D4963A
#66ADCB	D4966A
#66AERP	D4966A
#66ATTN	D4966A
#69ATTN	D4969A
#69ERP	D4969A
#69ERP8	D4969A
ACCACTL	IOSACCA
ACLOSE	D4969A
ACLOSE	$D69AXPS
ACLOSERT	RLOADER
AIIA	$SBAI
ALLSCB	NOSRCHK
ALPOSTED	EDXSVCX
AND1	EDXALU
AND1XX	EDXALU
AND2	EDXALU
AND2XX	EDXALU
AND4	EDXALU
AND4XX	EDXALU
ASMDEBUG	EDXINIT1
ATTACH	EDXSVCX
ATTACHX	EDXSVCX
BDCDWORD	EDXTIO
BDCWORD	EDXTIO
BFRCK	EDXALU
BFRCK	XPSSYS
BHXWORD	EDXTIO
BRACKN	$EDXTIO
BRANCH	EDXALU
BSCENTRY	BSCAM
BSCIA	BSCAM
BSCINIT	BSCINIT
BSCRETER	BSCAM
BSCGRET	BSCAM
CCBFIXA	EDXINIT
CDRVTA	$EDXDEF (TERMINAL)
CDRVTB	$EDXDEF (TERMINAL)
CGOTO	EDXALU
CIRCNT	CIRCBUFF
CIREND	CIRCBUFF
CIRESIZ	CIRCBUFF
CIRESTR	CIRCBUFF
CIRIN	CIRCBUFF
CIRSTR	CIRCBUFF
CKEXIT	SBCOM
CKEXITO	$SBCOM
CMDSETUP	EDXALU
CMDSETUP	XPSSYS
CMDTABLE	EDXSYS
CMPRSDME	RSLCMPR
CNTLBUSY	D4963A
CNTLEND	D4963A
CNTLEND	$D63AXPS
CONINSIA	EDXSTART
CTLXFER	EDXTIO
CTLXFER	NOTIO
CTLXFER	$TIOXPS
CURCTL	EDXTIO
CURCTL	NOTIO
CURCTL	$TIOXPS
DATEFMT	$EDXDEF (TIMER)
DATTN10	$D624XPS
DATTN10	D49624
DATTN10A	$D63AXPS
DATTN10A	D4963A
DATTN35	$D624XPS
DATTN35	D49624
DATTN40	$D624XPS
DATTN40	D49624
DATTN50	D4966A
DATTN50	$D66AXPS
DATTN50A	D4966A
DATTN51	$D66AXPS
DATTN51	D4966A
DATTN55A	$D66AXPS
DATTN55A	D4966A
DATT1024	DSKXPS
DATT1024	$D1024
DCBDWORD	EDXTIO
DCBRETRN	DISKIO
DCBWORD	EDXTIO
DCB3DATA	EDXTIO
DEBUGIT	BSCAM
DECSCAN	EDXTIO
DELRT	$DSKXPS
DELRT	DISKIO
DELSCB	NOSRCHK
DELSCBX	NOSRCHK
DEQ	EDXSVCX
DEQBSC	BSCAM
DEQHOST	TPCOM
DEQLBNK	NOLDRCHK
DEQT	NOTIO
DEQT	EDXTERMQ
DEQT	$TIOXPS
DETACH	EDXSVCX
DFLIH04	$DISKIO
DFLIH04	DSKXPS
DFLIH36A	$D624XPS
DFLIH36A	D49624
DFLIH40	D4963A
DFLIH40	$D63AXPS
DFLIH50	D49624
DFLIH50	$D624XPS
DFLIH54	D49624
DFLIH54	$D624XPS
DIIA	$SBDIDO
DINITDS1	EDXINIT1
DISKATTN	D49624
DISKBUFR	SEGINIT
DISKBUFX	DISKINIT
DISKCHN	DISKIO
DISKERR1	$DISKIO
DISKERR1	DSKXPS

DISKERR2	$DISKIO
DISKERR2	DSKXPS
DISKERR3	$DISKIO
DISKERR3	DSKXPS
DISKERR5	$DISKIO
DISKERR5	DSKXPS
DISKERR7	$DISKIO
DISKERR7	DSKXPS
DISKER13	DISKIO
DISKER13	$DSKXPS
DISKER17	$DISKIO
DISKER17	DSKXPS
DISKER18	$DISKIO
DISKER18	DSKXPS
DISKER5A	$DISKIO
DISKER5A	DSKXPS
DISKER5B	$DISKIO
DISKER6B	$DISKIO
DISKER6B	DSKXPS
DISKFLIH	$DISKIO
DISKFLIH	DSKXPS
DISKINT1	DISKINT1
DISKINTX	DSKINIT2
DISKIO	$DSKXPS
DISKIO00	DISKIO
DISKIO00	$DSKXPS
DISKIO32	D4966A
DISKIO40	D4966A
DISKPOST	$DISKIO
DISKPOST	DSKXPS
DISKPREP	DISKINIT
DISKRET	DISKIO
DISKRET2	DISKIO
DISKRW	DISKIO
DISKRW	$DSKXPS
DISKRW05	DISKIO
DKSCAC	$D63AXPS
DMDDB	$EDXDEF (DISK)
DMIPL	$EDXDEF (DISK)
DMVOL	$EDXDEF (DISK)
DMXTNT	$DISKIO
DMXTNT3	$DISKIO
DOIA	$SBDIDO
DQTERM	EDXTERMQ
DQTERMB	EDXTERMQ
DQTERMS	EDXTERMQ
DQTRMIN	EDXTERMQ
DSCKSUM	RSLCMPR
DSKCAC	D4963A
DSKCAC	$D63AXPS
DSKCHKA	NODSKCHK
DSKCHKD	NODSKCHK
DSKCHKAR	DISKIO
DSKCHKDR	DISKIO
DSKCHKER	DISKIO
DSKINITX	DSKINIT2
DSKINIT1	DSKINIT1
DSKPREP2	DISKINIT
DSKXRET1	DISKIO
DSKXRET2	DISKIO
DSKXRET3	DISKIO
D1024ATN	D1024
D1024FLG	$D1024
D1024R	D1024
D1024RET	D1024
D1024V1	$D1024
D4962IH1	D49624
D49624AT	$D49624
D4963AT	$D4963A
D4963ATN	D4963A
D4963AT1	D4963A
D4963IH1	D4963A
D4966AT	$D4966A
D4966ATN	D4966A
D4966B	D4966A
D4966B	$D66AXPS
D624ATTN	$D624XPS
D624ATTN	D49624
D624ERP	$D624XPS
D624ERP	D49624
D624INIT	$D624XPS
D624INIT	D49624
D63ATTN	$D63AXPS
D63ATTN	D4963A
D63ERP	$D63AXPS
D63ERP	D4963A
D63INIT	$D63AXPS
D63INIT	D4963A
D66AERP	$D66AXPS
D66AERP	D4966A
D66ATTN	$D66AXPS
D66ATTN	D4966A
D66INIT	DISKINIT
D69ATTN	$D69AXPS
D69ATTN	D4969A
D69DHPT	$D4969A
D69ERP	$D69AXPS
D69ERP	D4969A
D69ERP8	D4969A
D69ERP8	$D69AXPS
D69FLIH	$D4969A
EBBICVT	EDXTIO
EBBICVT	NOTIO
EBBICVT	$TIOXPS
EBBIE	EDXTIO
EBFLDBL	EBFLCVT
EBFLPTCH	EBFLCVT
EBFLSTD	EBFLCVT
EDXDEBUG	EDXINIT1
EDXFLAGS	EDXSYS
EDXFLEND	EDXFLOAT
EDXINITP	EDXINIT1
EDXLOOP	EDXALU
EDXLOOP2	EDXALU
ENABLED	EDXINIT1
ENDATTN	EDXTIO
ENDATTN	NOTIO
ENDATTN	$TIOXPS
ENDCODE	RLOADER
ENDINIT	EDXSTART

Supervisor Module Names (CSECTS)

ENQ	EDXSVCX
ENQLBNK	NOLDRCHK
ENQT	EDXTIO
ENQT	NOTIO
ENQT	EDXTERMQ
ENQT	$TIOXPS
EOR1	EDXALU
EOR1XX	EDXALU
EOR2	EDXALU
EOR2XX	EDXALU
EOR4	EDXALU
EOR4XX	EDXALU
ERREXIT3	RLOADER
ERRORLOG	DISKIO
EXCLOSE	IOSEXIO
EXFLIH	IOSEXIO
EXIO	IOSEXIO
EXIOCLEN	IOSEXIO
EXIOINIT	EXIOINIT
EXOPEN	IOSEXIO
EXPNDME	RSLCMPR
FADD000	EDXFLOAT
FADD000	NOFLOAT
FADD001	EDXFLOAT
FADD001	NOFLOAT
FADD010	EDXFLOAT
FADD010	NOFLOAT
FADD011	EDXFLOAT
FADD011	NOFLOAT
FADD100	EDXFLOAT
FADD100	NOFLOAT
FADD101	EDXFLOAT
FADD101	NOFLOAT
FADD110	EDXFLOAT
FADD110	NOFLOAT
FADD111	EDXFLOAT
FADD111	NOFLOAT
FDIV000	EDXFLOAT
FDIV000	NOFLOAT
FDIV001	EDXFLOAT
FDIV001	NOFLOAT
FDIV010	EDXFLOAT
FDIV010	NOFLOAT
FDIV011	EDXFLOAT
FDIV011	NOFLOAT
FDIV100	EDXFLOAT
FDIV100	NOFLOAT
FDIV101	EDXFLOAT
FDIV101	NOFLOAT
FDIV110	EDXFLOAT
FDIV110	NOFLOAT
FDIV111	EDXFLOAT
FDIV111	NOFLOAT
FH4963CT	RW4963ID
FIRSTACB	$EDXDEF (ADAPTER)
FIRSTBSC	$EDXDEF (BSCLINE)
FIRSTCCB	$EDXDEF (TERMINAL)
FLDCLEAR	EDXTIO
FLEBDBL	EBFLCVT
FLEBSTD	EBFLCVT
FLIHPOST	$D4969A
FLIHPOST	DSKXPS
FLOATERR	EDXFLOAT
FLOATERR	NOFLOAT
FLRND	EBFLCVT
FLTCONV	EDXFLOAT
FLTCONV	NOFLOAT
FLTPATCH	EDXFLOAT
FMPY000	EDXFLOAT
FMPY000	NOFLOAT
FMPY001	EDXFLOAT
FMPY001	NOFLOAT
FMPY010	EDXFLOAT
FMPY010	NOFLOAT
FMPY011	EDXFLOAT
FMPY011	NOFLOAT
FMPY100	EDXFLOAT
FMPY100	NOFLOAT
FMPY101	EDXFLOAT
FMPY101	NOFLOAT
FMPY110	EDXFLOAT
FMPY110	NOFLOAT
FMPY111	EDXFLOAT
FMPY111	NOFLOAT
FREEMAIN	RLOADER
FREERETN	STORMGR
FREESTGC	STORMGR
FSUB000	EDXFLOAT
FSUB000	NOFLOAT
FSUB001	EDXFLOAT
FSUB001	NOFLOAT
FSUB010	EDXFLOAT
FSUB010	NOFLOAT
FSUB011	EDXFLOAT
FSUB011	NOFLOAT
FSUB100	EDXFLOAT
FSUB100	NOFLOAT
FSUB101	EDXFLOAT
FSUB101	NOFLOAT
FSUB110	EDXFLOAT
FSUB110	NOFLOAT
FSUB111	EDXFLOAT
FSUB111	NOFLOAT
FULLMSG	FULLMSG
GETCNT	EDXALU
GETDDB	SBCOM
GETMAIN	RLOADER
GETPAR3	EDXALU
GETPAR3	XPSSYS
GETRETF	STORMGR
GETRETN	STORMGR
GETVAL2	EDXTIO
GSTVAL2	NOTIO
GSTVAL2	$TIOXPS
GETVAL4	EDXTIO
GSTVAL4	NOTIO
GSTVAL4	$TIOXPS
GFQCB	RLOADER
GOTOTABL	RLOADER
GPIB	$EDXDEF

GTIMDATE	EDXTIMER
GTIMDATE	EDXTIMR2
HCACK0	TPCOM1
HCACK1	TPCOM1
HCDLEETX	TPCOM1
HCDLESTX	TPCOM1
HCENQCC	TPCOM1
HCEOT	TPCOM1
HCFLAGS	TPCOM1
HCID	TPCOM1
HCLENHS	TPCOM1
HCLENPT	TPCOM1
HCNACK	TPCOM1
HCRESNE	TPCOM1
HCRESNI	TPCOM1
HCRESNI	TPCOM1
HCRESP0	TPCOM1
HCRESP1	TPCOM1
HCRETCD	TPCOM1
HCSMID	TPCOM1
HCSTBYTE	TPCOM1
HCTDBUFF	TPCOM1
IAACCA	$IOSACCA
IAACCATR	$ACCATRC
IACEND	$IOSACCA
IACHKCC	$IOSACCA
IADELAY	$IOSACCA
IAEXIOTR	EXIOTRC
IAGPIB	$IOSGPIB
IAMQCB	CDIDQCB
IAPROC	$IOS2741
IASCII	$IO4975A
IAS1S1	$IOSS1S1
IATTY	$IOSTTY
IA2741	$IOS2741
IA4013	$IOS4013
IA4973	$IOS4974
IA4974	$IOS4974
IA4975	$IOS4974
IA4978	$IOS4979
IA4979	$IOS4979
IA4980	$IOS4979
IA5219	$IOS4974
IA5224	$IOS4974
IA5225	$IOS4974
IDCBRES	TPCOM
IDCBSCSS	TPCOM
IDCBSIO	TPCOM
IDCBSIO	TPCOM1
IDCBSIOD	TPCOM1
IDCBRES	TPCOM1
IDCBSCSS	TPCOM1
IDCBTOD	TPCOM1
IDSKIO99	$DISKIO
IDSKIO99	DSKXPS
IDSKPOST	$DISKIO
IDSKPOST	DSKXPS
ID4963IH	RW4963ID
IFFLOAT	EDXFLOAT
IFFLOAT	NOFLOAT
IFFLOATL	EDXFLOAT
IFFLOATL	NOFLOAT
INITEXIT	EDXINIT1
INITFEAT	EDXINIT1
INITTASK	EDXSTART
INIT4013	INIT4013
INIT4978	INIT4978
INTIME	EDXTIMER
INTIME	EDXTIMR2
INTIMEX	EDXTIMER
INTIMEX	EDXTIMR2
IOACPOST	NOACCATR
IOACPOST	ACCATRC
IOACPRE	NOACCATR
IOACPRE	ACCATRC
IOERROR	IOSACCA
IOEXPOST	NOEXIOTR
IOEXPOST	EXIOTRC
IOEXPRE	NOEXIOTR
IOEXPRE	EXIOTRC
IOGPIB	$IOSGPIB
IOLOAD	IOLOADER
IOR1	EDXALU
IOR1XX	EDXALU
IOR2	EDXALU
IOR2XX	EDXALU
IOR4	EDXALU
IOR4XX	EDXALU
IOSPCLOS	IOSPOOL
IOSPCMD	$IOSPOOL
IOSPDQT	IOSPOOL
IOSPEND	$IOSPOOL
IOSPNQT	IOSPOOL
IOSPTBL	$IOSPOOL
IOSPWR	IOSPOOL
IOSRETN	IOSACCA
IOSTABLE	IOSTABLE
IOSTRT	IOSACCA
IOS1S1	IOSS1S1
IOS3101D	IOS3101
IOS4979M	IOS4979
IOUNLOAD	IOLOADER
IOWRTCON	IOSACCA
IO1024I	IO1024
IO1024RT	DISKINIT
IO3101	IOS3101
IO4974	IOS4974
IO4974	NOTIO
IO4974	$TIOXPS
IO4975A	IOS4975A
IO4979	IOS4979
IO4979	NOTIO
IO4979	$TIOXPS
IPLENDED	EDXSTART
IPLMSG	DISKINIT
IPLMSG	EDXINIT1
KBTASK	EDXTIO
KBTASK	NOTIO
KBTASK	$TIOXPS
LCB	SYSLOG

Supervisor Module Names (CSECTS)

LCBA	EDXSYS
LCMDKEY	RLOADER
LCMDTGT	RLOADER
LEX	EDXSVCX
LOADBUFR	LOADBUFR
LOADEXIT	RLOADER
LOADFHFL	RLOADER
LOADFLAG	NOSRMGR
LOADINIT	LOADINIT
LOADINIT2	LOADINIT2
LOADORG	RLOADER
LOADPGM	RLOADER
LOADPGM0	RLOADER
LOADQCB	RLOADER
LOADTASK	RLOADER
LOADTERM	PWRM80
LPGMXPA	RLOADER
LPGMXPB	RLOADER
LPRINTER	$EDXDEF (TERMINAL)
LP0	RLOADER
MAPEND	$EDXDEF (SYSTEM)
MECBLST	$EDXDEF (SYSTEM)
MESSAGE	FULLMSG
MESSAGE	NOTIO
MESSAGE	$TIOXPS
MFARMU	$EDXDEF (ADAPTER)
MINMSG	MINMSG
MOVEXP	EDXALU
MOVFP4	EDXFLOAT
MOVFP4	NOFLOAT
MOVFP8	EDXFLOAT
MOVFP8	NOFLOAT
MOV1	EDXALU
MOV1C	EDXALU
MOV2	EDXALU
MOV2C	EDXALU
MOV4	EDXALU
MOV4C	EDXALU
MPRTR1	$EDXDEF (TERMINAL)
MPRTR2	$EDXDEF (TERMINAL)
MPRTR3	$EDXDEF (TERMINAL)
MSGBUF	FULLMSG
MSGBUFR	EDXSTART
MSGMOD	FULLMSG
MSGMOD	MINMSG
NAKMSG	$EDXTIO
NEXTERM	TERMINIT
NOTDEF	$EDXTIO
NOTPWRON	$IOS4979
NSRMGR	NOSRMGR
NXTCOMD	EDXTIO
OPNCLS	D4969A
OVLAREA0	SEGINIT
OVLEND	DISKINIT
OVLSTART	SEGINIT
PACK	RSLCMPR
PARTFLAG	NOSRMGR
PASS#2	EDXINIT1
PCHKIIP	EDXSTART
PCHKLSB	EDXSTART
PCHKSIA	EDXSTART
PGMCHECK	EDXINIT1
PGMCHK	XPSSYS
PIIABIT	$SBPI
PIIAG17	$SBPI
PIIALEX	$SBPI
PNTREND	EDXSVCX
POST	EDXSVCX
POSTWTM	SWAITM
PREPIDCB	DISKINIT
PRINTIME	EDXTIMER
PRINTIME	EDXTIMR2
PRSKSP	EDXTIO
PRSKSP	NOTIO
PRSKSP	$TIOXPS
PRTEXT	EDXTIO
PRTEXT	NOTIO
PRTEXT	$TIOXPS
PRTNUM2	EDXTIO
PRTNUM2	NOTIO
PRTNUM2	$TIOXPS
PRTNUM2S	EDXTIO
PRTNUM4	EDXTIO
PRTNUM4	NOTIO
PRTNUM4	$TIOXPS
PRTNUM4S	EDXTIO
PSTUSER	DISKIO
PWRAM80	PWRM80
PWRTEST	PWRM80
QIO	QIO
QUESTION	EDXTIO
QUESTION	NOTIO
QUESTION	$TIOXPS
QUTERM	EDXTERMQ
QUTERMIN	EDXTERMQ
RADDRESS	TPCOM1
RDACCA	IOSACCA
RDCB1CNT	TPCOM1
RDCB1	TPCOM1
RDCB2	TPCOM1
RDCB3	TPCOM1
RDCB4	TPCOM1
RDCB5	TPCOM1
RDCB6	TPCOM1
RDCB7	TPCOM1
RDCB8	TPCOM1
RDCB9	TPCOM1
RDGPIB	$IOSGPIB
RDS1S1	IOSS1S1
RDTEXT	EDXTIO
RDTEXT	NOTIO
RDTEXT	$TIOXPS
RDTEXTL	EDXTIO
RDTEXTL	NOTIO
RDTEXTL	$TIOXPS
RDTTY	IOSTTY
RD2741	IOS2741
RD4013	IOS4013
RECKSUM	RSLCMPR
RETURN	EDXSYS

RLENGTH	TPCOM1
SAI	SBAI
SAIA	SBAI
SAIS	SBAI
SAIX	SBAI
SAO	$SBAO
SAOA	$SBAO
SAOX	$SBAO
SATTACH	EDXSVCX
SAVERET	EDXSVCX
SAVEXIT	EDXSVCX
SAV222CR	EDXALU
SAV224CR	EDXALU
SAV424CR	EDXALU
SAV444CR	EDXALU
SAX222	EDXALU
SA222	EDXALU
SA222C	EDXALU
SA424	EDXALU
SA424C	EDXALU
SBERR	SBCOM
SBIODDB	$EDXDEF (SENSORIO)
SBIOINIT	SBIOINIT
SCALL	EDXALU
SCONTINU	EDXALU
SCWAIT	EDXSVCX
SDEQ	EDXSVCX
SDETACH	EDXSVCX
SDI	$SBDIDO
SDIA	$SBDIDO
SDIS	$SBDIDO
SDIX	$SBDIDO
SDO	$SBDIDO
SDOA	$SBDIDO
SDOLOOP	EDXALU
SDOP	$SBDIDO
SDOS	$SBDIDO
SDOX	$SBDIDO
SD222	EDXALU
SD222C	EDXALU
SD422CR	EDXALU
SD422R	EDXALU
SD424	EDXALU
SD424C	EDXALU
SEGINIT	SEGINIT
SEGREGE	NOSTRMGR
SEGREGE	STORMGR
SELB01	EDXSVCX
SENQ	EDXSVCX
SETBUSY	EDXSVCX
SETBUSY	XPSSYS
SETCLOCK	EDXTIMER
SETCLOCK	EDXTIMR2
SETIMER	EDXTIMER
SETIMER	EDXTIMR2
SETREADY	EDXSVCX
SFLIST	$EDXTIO
SFTKSIA	EDXSTART
SHL1	EDXALU
SHL1XX	EDXALU
SHL2	EDXALU
SHL2XX	EDXALU
SHL4	EDXALU
SHL4XX	EDXALU
SHR1	EDXALU
SHR1XX	EDXALU
SHR2	EDXALU
SHR2XX	EDXALU
SHR4	EDXALU
SHR4XX	EDXALU
SM222	EDXALU
SM222C	EDXALU
SM424	EDXALU
SM424C	EDXALU
SPOST	EDXSVCX
SRADDR	EDXINIT1
SRESETEV	EDXSVCX
SRETURN	EDXALU
SSX222	EDXALU
SS222	EDXALU
SS222C	EDXALU
SS424	EDXALU
SS424C	EDXALU
START	EDXINIT
STARTFLG	EDXINIT
STARTPGM	EDXSTART
START1	EDXINIT1
STATUS	IOSACCA
STESTIN	$DBUGNUC
STESTIN1	$DBUGNUC
STESTOUT	$DBUGNUC
STKINIT	SEGINIT
STMTBLE	STORMGR
STOP	RLOADER
STOPC5	BSCINIT
STOPDA	INIT4980
STOPDB	INITMFA
STOPDC	INITMFA
STOPDD	INITMFA
STOPDE	INITMFA
STOPDF	INITMFA
STOPEB	INIT4980
STOPEC	INITADAP
STOPED	INITADAP
STOPEE	INITADAP
STOPEF	INITADAP
STOPFB	EDXSVCX
STOPFC	EDXSVCX
STOPFD	EDXSVCX
STOPFE	EDXSVCX
STOPF0	SEGINIT
STOPF1	XPSINIT
STOPF2	XPSINIT
STOPF3	XPSINIT
STOPF4	XPSINIT
STOPF5	XPSINIT
STOPF6	XPSINIT
STOPF7	XPSINIT
STOPF8	XPSINIT
STOPF9	XPSINIT

Supervisor Module Names (CSECTS)

STOPTASK	RLOADER
STOREMAP	$EDXDEF
STORINIT	STORINIT
STP	TPCOM
STPTASK1	EDXSVCX
STPTASK2	EDXSVCX
SUPEXIT	EDXSVCX
SUPEXIT	XPSSYS
SUPEXTRL	EDXSVCX
SUPLEX	EDXSVCX
SUPLVLX0	EDXSVCX
SUPTBL	EDXSVCX
SVC	EDXSVCX
SVC	XPSSYS
SVCA	EDXSVCX
SVCAKR	EDXSYS
SVCBF	$EDXDEF (SYSTEM)
SVCBFEND	$EDXDEF (SYSTEM)
SVCBFIN	EDXSYS
SVCBFOUT	EDXSYS
SVCBOTH	EDXSVCX
SVCFLAGS	EDXSYS
SVCFLAGS	EDXSVCX
SVCHNDLR	EDXSVCX
SVCI	EDXSVCX
SVCIAKR	EDXSYS
SVCIAR	EDXSYS
SVCIBFOF	EDXSVCX
SVCIIAR	EDXSYS
SVCILSB	EDXSYS
SVCILSR	EDXSYS
SVCIR0	EDXSYS
SVCIR1	EDXSYS
SVCIR2	EDXSYS
SVCIR3	EDXSYS
SVCIR4	EDXSYS
SVCIR5	EDXSYS
SVCIR6	EDXSYS
SVCIR7	EDXSYS
SVCLSB	EDXSYS
SVCLSR	EDXSYS
SVCLT	EDXSYS
SVCL1	EDXSYS
SVCL2	EDXSYS
SVCL3	EDXSYS
SVCPARMS	EDXSYS
SVCRTRN	EDXSYS
SVCR0	EDXSYS
SVCR1	EDXSYS
SVCR2	EDXSYS
SVCR3	EDXSYS
SVCR4	EDXSYS
SVCR5	EDXSYS
SVCR6	EDXSYS
SVCR7	EDXSYS
SVCTRC	EDXSVCX
SVC0	EDXSVCX
SVC1	EDXSVCX
SVC2	EDXSVCX
SWAIT	EDXSVCX
SX224CR	EDXALU
SX224R	EDXALU
SX444	EDXALU
SX444C	EDXALU
SYSCOM	EDXSYS
S1S1	$EDXDEF (TERMINAL)
S1S1DIAG	S1S1INIT
TAPEEND	$D4969A
TAPEEND	DSKXPS
TAPEEND	$D4969A
TAPEINIT	TAPEINIT
TAPEIO	D4969A
TAPEIO00	D4969A
TAPEIO00	$D69AXPS
TAPEIO99	DSKXPS
TAPEIO99	$D4969A
TAPE060	DISKIO
TASKWD	EDXSTART
TERMDEFS	$EDXDEF (TERMINAL)
TERMERRX	TERMINIT
TERMINID	TERMINIT
TERMINT	EDXTIO
TERMOPEN	TERMOPEN
TERMOUT	EDXTIO
TIMERDDB	$EDXDEF (TIMER)
TIMER0	$EDXDEF (TIMER)
TIMER0	EDXTIMR2
TIMER0IA	EDXTIMER
TIMER0IA	EDXTIMR2
TIMER1	$EDXDEF (TIMER)
TIMER1IA	EDXTIMER
TIMRINIT	TIMRINIT
TIMRINIT	CLOKINIT
TIMRLSB	EDXTIMR2
TLENGTH	TPCOM1
TLSB	EDXSVCX
TPIABUSY	$D4969A
TPIABUSY	DSKXPS
TPIABUSY	$D4969A
TPINIT	TPINIT
TPSTATS	TPCOM
TRASCII	ASCIITAB
TRASCII	TRASCII
TRCRSP	CRSPTAB
TRCRSP	TRASCII
TREBASC	EBASCII
TREBASC	TREBASC
TREBCD	EBCDTAB
TREBCD	TREBCD
TSOPEN	D4969A
UNCHAIN	XPSSYS
UNCHAIN	EDXSVCX
UNCHAKR1	EDXSYS
UNCHSAV6	EDXSYS
UNPACK	RSLCMPR
USER	EDXALU
USERFLAG	INITMODS
USERMOD	INITMODS
USERPOST	DISKIO
VARYDSCB	DISKIO

VARYDSCB	$DISKIO
VARYEXIT	DISKIO
VARYEXIT	$DISKIO
VARYOFF	DISKIO
VARYOFF	$DISKIO
VARYON	DISKIO
VARYON	$DISKIO
VARYQCB	DISKIO
VARYQCB	$DISKIO
VARYWORD	DISKIO
VARYWORD	$DISKIO
VCTBL	IOSTABLE
VRYBUF	TAPEBUFR
VRY4966	$D4966A
VRY4969	$D4969A
WAIT	EDXSVCX
WAITIMER	EDXTIMER
WAITIMER	EDXTIMR2
WAITM	SWAITM
WAITMR	SWAITM
WDCB1	TPCOM1
WDCB2	TPCOM1
WDCB3	TPCOM1
WDCB3CHN	TPCOM1
WDCB4	TPCOM1
WDCBCHN	TPCOM1
WDCB5	TPCOM1
WHATIME	EDXTIMER
WHATIME	EDXTIMR2
WHATMSG	$EDXTIO
WRACCA	IOSACCA
WRGPIB	$IOSGPIB
WRPRSTAT	IOS4975A
WRS1S1	IOSS1S1
WRTTY	IOSTTY
WR2741	IOS2741
WR4013	IOS4013
WR4975A	IOS4975A
WTMERCD	EDXSVCX
XLATE	RSLCMPR
XPSBAL1	EDXSVCX
XPSBAL6	EDXSVCX
XPSBAL7	EDXSVCX
XPSBR	EDXSVCX
XPSDEBUG	EDXSVCX
XPSHEADR	SEGINIT
XPSINITX	XPSINIT
XPSRET2	EDXSVCX
XPSRET3	EDXSVCX
XPSSTK	$EDXDEF (SYSTEM)
XPSSTKE	$EDXDEF (SYSTEM)
XPSSVCI	XPSSYS
XPSSVCI	EDXSVCX
XPSSVCIX	EDXSVCX
XPSTABLE	XPSTABLE
XPSTBL	XPSTABLE
XPSTSIZE	XPSTABLE
X21CLR	BSCX21
X21CON	BSCX21

Notes

Appendix E. Work Sheets

This appendix contains the work sheets you need to select the required support for your operating system.

- Use WORK SHEET 1 to estimate the overall size of your supervisor. Use work sheet 4 to estimate the size of portions of your supervisor.
- Use WORK SHEET 2 to define the characteristics and partition structure of processor storage and the I/O devices attached to your Series/1.
- Use WORK SHEET 3 to define the software features you require to support the defined I/O devices and the EDX-related products you include in your system.
- Use WORK SHEET 4 to estimate the size of those portions of the supervisor that in partitions other than partition one and the amount of storage required by programs to execute within a partition.

The work sheets are provided for you to use with Chapter 4, "Select Your Required Support" on page IS-39. Tear out or copy the work sheets you require, and complete them as you select the system features you need for your system. We recommend making a copy of each work sheet so that you will have the originals if you need to generate a system in the future.

Work Sheets

Work Sheet 1

To estimate the overall size of your supervisor, use work sheet 1. Work sheet 1 contains three columns: Support, Resident, and Initialization.

Support

This column lists the I/O devices you can attach to the Series/1. Read down the list and choose the specific devices you are attaching to your Series/1; cross out the devices that are not part of your system, along with the corresponding storage amounts. The Resident column and the Initialization column, if applicable, show the amount of storage required by the supervisor to support a specific device.

Resident

This column contains the amount of storage the supervisor requires to support each device. These numbers are the resident program sizes and are expressed in the number of decimal bytes.

Initialization

This column contains the amount of storage the initialization routine requires at IPL time for a specific device. These numbers are expressed in the number of decimal bytes.

Supervisor Size

To estimate the size of your supervisor, total the Resident column and round this total to the next multiple of 256. The resulting figure is the estimated size of your supervisor program that will reside in storage during system execution. Allow from three to five per cent more storage to provide for error correction. This estimate will be reasonably close to your actual configuration; to get the actual size, perform a system generation of your supervisor. The actual size is the address (in hexadecimal) of EDXINIT in the $XPSLINK output.

Work Sheet 1 *(continued)*

Support	Resident	Initialization
Basic supervisor		
with address translator	8058	614
(sum of MAXPROG values) * 8 + 32	( )	
+ #IABUF (defaults to 20) * 8	( )	
+ #XPSSTK (defaults to 20) * 6	( )	
+ #MECBLST (defaults to 20) * 2	( )	
Disk, diskette, or tape		
Disk(ette) basic	1912	3420
+200 per unit	( )	
+ 47 per performance volume	( )	
+128 per I/O task defined	( )	
4962/4964	1466	
4963/4967/DDSK-30	692	
4963 with fixed-head		526
+178 per unit	( )	
4965/4966	1652	
1024 Byte/Sector Support	428	
1024 Byte/Sector IPL Support	2596	
4968/4969	4920	1706
+130 per unit	( )	
+128 per I/O task defined	( )	
Terminals		
Basic	5664	1182
MFA (feature 1310)		1036
+50 per adapter	( )	
ALPA (feature 1250)		2478
+50 per adapter	( )	
SMIO (feature 5640)		2478
+50 per adapter	( )	
4979/4978/4980	2640	1812
+488 per 4978/4979/4980	( )	
4973/4974/4975/5219/5224/5225	744	
+550 per 4973 or 4974	( )	
+562 per 4975	( )	
+562 per 5219	( )	
+562 per 5224	( )	
+562 per 5225	( )	
4013 type devices	506	182
+458 per device	( )	

Figure 23 (Part 1 of 3). Supervisor Storage Requirements.

Work Sheet 1 *(continued)*

Support	Resident	Initialization
Virtual terminals	650	
+470 per terminal	( )	
Series/1 to Series/1	1264	274
+624 per device	( )	
GPIB	734	
+574 per adapter	( )	
Basic for		
any TTY, 2741/PROC, non-3101B, ACCA	648	
+514 if ASCII/EBCDIC conversion	( )	
+514 if mirror image ASCII/EBCDIC conversion	( )	
Teletypewriter	740	
+470 per teletypewriter	( )	
2741/PROC	1524	
+594 per 2741/PROC	( )	
+532 if correspondence code	( )	
+522 if EBCD code	( )	
ACCA ASCII Terminals no MFA	1848	
+530 per terminal	( )	
3101 (block mode)	2250	
+724 per device	( )	
Spool		
Basic (in supervisor)	2104	
+(460*max active jobs)+32	( )	
Basic (in user partition)	5120	
+Y+Z	( )	
where:		
Y=264*max active jobs		
Z=(34*max jobs)		
+(2*no. of sectors/100		
+(20*no. of writers)		
Each output writer (in user partition)	3840	
Timers		
4955	1308	354
4952/4954/4956	1264	188

Figure 23 (Part 2 of 3). Supervisor Storage Requirements.

Work Sheet 1 *(continued)*

Support	Resident	Initialization
Binary synchronous access method		
Any BSC device	3884	430
In addition:		
+ 22 per multiline feature	( )	
+136 per line	( )	
X.21 Facility	1760	
Host Communication Facility	2238	364
Sensor-based input/output		
Basic	144	180
Analog input	702	
+48 for first AI group	( )	
+16 for each additional group	( )	
Analog output	90	
+16 per AO	( )	
Digital input and output	1014	
+38 per DI group	( )	
+16 per DO group in 4982	( )	
+38 per DO group in IDIO	( )	
Process interrupt	168	
+156 per PI group	( )	
EXIO control		
Basic	830	66
+(32+x(16+n)) per device	( )	
where:		
x=maximum number of DCBs		
n=number of residual status bytes transferred		
Error logging		
included	362	
not included	26	
Program/machine check log	352	
Relocating loader	4044	1986
Floating point		
included	772	
not included	6	
Support of GETEDIT/PUTEDIT	1628	
Queue processing	326	
$DEBUG	522	
Supervisor patch area	256	

Figure 23 (Part 3 of 3). Supervisor Storage Requirements.

Notes:

1. The above numbers include 128 bytes per terminal for the optional keyboard task (ATTN=YES).

2. Teletypewriter, ACCA, 2741, and 4013 terminals require basic ASCII support.

Work Sheet 1 *(continued)*

This page intentionally left blank.

Work Sheet 2

Work sheet 2 helps you define your processor storage and I/O devices. The work sheet contains the system definition statements used in this step. Each definition statement is presented in the form:

```
label      STATEMENT    OPERAND
blank      SYSTEM       MAXPROG=(__,__,__,__,__,__,__,__),
                        PARTS=(__,__,__,__,__,__,__,__),
                        DATEFMT=____,
                        IABUF=____,
                        XPSSTAK=__,
                        COMMON=(__,__,__,__,__,__,__,__),
                        INITPRT=____,
                        INITMOD=(____,____,____,____),
                        MECBLST=___
```

In addition, you can use specific definition statements to define different devices. Depending upon the device you are defining, the system requires different operands. A sample statement is shown for each device type.

The right-hand column contains a key that indicates if an operand is required or optional and specifies the default. A detailed description of the statements, along with the syntax and operand definitions, is provided in Appendix A, "System Definition Statements" on page IS-145.

Work Sheet 2 *(continued)*

This page intentionally left blank.

IBM Series/1 Event Driven Executive

Work Sheet 2—Installation and System Generation

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Storage definition				
blank	SYSTEM	MAXPROG=	(______, ______, ______, ______, ______, ______, ______, ______),	Optional-defaults to 10
		PARTS=	(______, ______, ______, ______, ______, ______, ______, ______),	Optional-defaults to 32K
		DATEFMT=	______,	Optional-defaults to MMDDYY
		IABUF=	______,	Optional-defaults to 20
		XPSSTK=	______,	Optional-defaults to 20
		COMMON=	(______, ______, ______, ______, ______, ______, ______, ______),	Optional-defaults to EDXSYS
		INITPRT=	______,	Optional-defaults to 1
		INITMOD=	(______, ______, ______, ______)	Optional
		MECBLST=	______	Optional
Tape definition				
blank	TAPE	DEVICE=**4969,**		Required
		ADDRESS=	______,	Required
		DENSITY=	______,	Optional-defaults to 1600
		LABEL=	______,	Optional-defaults to SL
		ID=	______,	Required
		TASK=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO
blank	TAPE	DEVICE=**4968,**		Required
		ADDRESS=	______,	Required
		DENSITY=	______,	Optional-defaults to 1600
		LABEL=	______,	Optional-defaults to SL
		ID=	______,	Required
		TASK=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO
Disk(ette) definition				
blank	DISK	DEVICE=**4962,**		Required
		ADDRESS=	______,	Required
		VOLNAME=	(______, ______, ______, ______),	Optional
		TASK=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO

IBM Series/1 Event Driven Executive

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Disk(ette) definition (continued)				
blank	DISK	DEVICE=**4963,**		Required
		ADDRESS=	______,	Required
		VOLNAME=	(______, ______, ______, ______),	Optional
		TASK=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO
blank	DISK	DEVICE=**4967,**		Required
		ADDRESS=	______,	Required
		VOLNAME=	(______, ______, ______, ______),	Optional
		TASK=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO
blank	DISK	DEVICE=**DDSK-30**		Required
		ADDRESS=	______,	Required
		VOLNAME=	(______, ______, ______, ______),	Optional
		TASK=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO
			______,	(Specify YES if last TAPE
			______,	or DISK statement)
blank	DISK	DEVICE=**4964**		Required
		ADDRESS=	______,	Required
		TASK=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO
blank	DISK	DEVICE=**4965**		Required
		ADDRESS=	______,	Required
		TASK=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO
blank	DISK	DEVICE=**4966**		Required
		ADDRESS=	______,	Required
		TASK=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO

IBM Series/1 Event Driven Executive

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Timer definition				
blank	TIMER	ADDRESS=	______	Required
Adapter definition				
label	ADAPTER	ADDRESS=	______	Required
		TYPE=**MFA**,		Required
		DEVICES=	(______, ______, ______, ______),	Required
		END=	______	Optional-defaults to NO
label	ADAPTER	ADDRESS=	______	Required
		TYPE=**ALPA**,		Required
		DEVICES=	(______, ______, ______, ______, ______, ______, ______, ______),	Required
		END=	______	Optional-defaults to NO
label	ADAPTER	ADDRESS=	______	Required
		TYPE=**SMIO**,		Required
		DEVICES=	(______, ______, ______, ______, ______, ______, ______, ______),	Required
		END=	______	Optional-defaults to NO (Specify YES if last ADAPTER statement)
			______,	
Terminal definition				
Virtual Terminals				
label	TERMINAL	ADDRESS=**VIRT**		Required
		ADDRESS=	______,	Required-enter label of other virtual terminal
		LINSIZE=	______,	Optional-defaults to 132
		SYNC=	______,	Optional-defaults to NO
		END=	______	Optional-defaults to NO

IBM Series/1 Event Driven Executive

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Terminal definition (continued)				
Virtual Terminals (continued)				
label	TERMINAL	ADDRESS=**VIRT**		Required
		ADDRESS=	_______,	Required-enter label of other virtual terminal
		LINSIZE	_______,	Optional-defaults to 132
		SYNC=	_______,	Optional-defaults to NO
		END=	_______	Optional-defaults to NO (Specify YES is last TERMINAL statement)
Series/1=Series/1				
label	TERMINAL	DEVICE=**S1S1**		Required
		ADDRESS=	_______,	Required
		LINSIZE=	_______,	Optional-defaults to 80
		CHKSUM=	_______,	Optional-defaults to 16
		END=	_______,	Optional-defaults to NO (Specify YES is last TERMINAL statement)
GPIB (General Purpose Interface Bus)				
label	TERMINAL	DEVICE=**GPIB**		Required
		ADDRESS=	_______,	Required
		CODTYPE=	_______,	Optional-defaults to ASCII
		LINSIZE=	_______,	Optional-defaults to 80
		ATTN=	_______,	Optional-defaults to YES
		SCREEN=	_______,	Optional-defaults to NO (YES/ROLL or NO)
		PART=	_______,	Optional-defaults to partition 1
		SPOOL=	_______,	Optional-defaults to NO
		END=	_______,	Optional-defaults to NO (Specify YES if last TERMINAL statement)

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Terminal definition (continued)				
Processor to Processor Communications				
label	TERMINAL	DEVICE=**PROC**		Required
		ADDRESS=	______,	Required (in hexadecimal)
		CODTYPE=	______,	Optional-defaults to EBCDIC (DRSP/EBCD)EBCDIC)
		LINSIZE=	______,	Optional-defaults to 130
		CRDELAY=	______,	Optional-defaults to (PROMPT,300000) (PROMPT,n/SP5100,n/DELAY,n)
		BITRATE=	______,	Optional-defaults to 9600
		RANGE=	______,	Optional-HIGH/LOW-defaults to HIGH
		CR=	______,	Optional-defaults to 5B
		LF=	______,	Optional-defaults to 5B
		END=	______,	Optional-defaults to NO (Specify YES if last TERMINAL statement)

IBM **Series/1 Event Driven Executive**

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Terminal definition (continued)				
2741 Terminal				
label	TERMINAL	DEVICE=2741,		Required
		ADDRESS=	______,	Required
		PAGSIZE=	______,	Optional-defaults to 66
		LINSIZE=	______,	Optional-defaults to 130
		CODTYPE=	______,	Optional-defaults to EBCD (EBCD/CRSP)
		TOPM=	______,	Optional-defaults to 0
		BOTM=	______,	Optional-defaults to 65
		LEFTM=	______,	Optional-defaults to 0
		RIGHTM=	______,	Optional-defaults to 129
		OVFLINE=	______,	Optional-defaults to NO
		LINEDEL=	______,	Optional-(2 digit hex char)-defaults to A0
		CHARDEL=	______,	Optional-(2 digit hex char)-defaults to 5D
		CRDELAY=	______,	Optional-defaults to 0
		ECHO=	______,	Optional-defaults to YES
		BITRATE=	______,	Optional-defaults to 134 bps
		ADAPTER=	______,	Optional-(specify SINGLE)
		CR=	______,	Optional-defaults to 5B
		LF=	______,	Optional-defaults to 5B
		SCREEN=	______,	Optional-defaults to NO (YES or ROLL/NO)
		PART=	______	Optional-defaults to partition 1
		SPOOL=		Optional-defaults to NO
		END=		Optional-defaults to NO (Specify YES if last TERMINAL statement

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Terminal definition (continued)				
4013 Graphics Terminal				
label	TERMINAL	DEVICE=**4013,**		Required
		PAGSIZE=	________,	Optional-defaults to 35
		LINSIZE=	________,	Optional-defaults to 72
		CODTYPE=	________,	Optional-specify ASCII
		TOPM=	________,	Optional-defaults to 0
		BOTM=	________,	Optional-defaults to 34
		LEFTM=	________,	Optional-defaults to 0
		RIGHTM=	________,	Optional-defaults to 71
		OVFLINE=	________,	Optional-defaults to NO
		LINEDEL=	________,	Optional-(2 digit hex char)-defaults to 7F
		CHARDEL=	________,	Optional-(2 digit hex char)-defaults to 08
		CRDELAY=	________,	Optional-defaults to 0
		ECHO=	________,	Optional-defaults to YES
		CR=	________,	Optional-defaults to 0D
		LF=	________,	Optional-defaults to 0A
		SCREEN=	________,	Optional-defaults to NO (YES/ROLL or NO)
		DI= / DO= / PI=	________,	Required (one of the three)
		PART=	________,	Optional-defaults to partition 1
		END=	________	Optional-defaults to NO (Specify YES if last TERMINAL statement)

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Terminal definition (continued)				
4973/4974 Printers				
label	TERMINAL	DEVICE=	______,	Required (specify 4973 or 4974)
		ADDRESS	______,	Required
		PAGSIZE=	______,	Optional-defaults to 66
		LINSIZE=	______,	Optional-defaults to 132
		TOPM=	______,	Optional-defaults to 3
		BOTM=	______,	Optional-defaults to 63
		LEFTM=	______,	Optional-defaults to 0
		RIGHTM=	______,	Optional-defaults to 131
		OVFLINE=	______,	Optional-defaults to NO
		SPOOL=	______,	Optional-defaults to NO
		END=	______,	Optional-defaults to NO (Specify YES if last TERMINAL statement)
4975 Printer				
label	TERMINAL	DEVICE=**4975**		Required
		ADDRESS=	______,	Required
		PAGSIZE=	______,	Optional-defaults to 66
		LINSIZE=	______,	Optional-defaults to 132
		CHARSET=	______,	Optional-defaults to USCA
		TOPM=	______,	Optional-defaults to 3
		BOTM=	______,	Optional-defaults to 63
		LEFTM=	______,	Optional-defaults to 0
		RIGHTM=	______,	Optional-defaults to 131
		OVFLINE=	______,	Optional-defaults to NO
		BITRATE=	______,	Optional-defaults to 1200 bps
		LMODE=	______,	Optional-defaults to LOCAL
		ADAPTER=**MFA**		Required
		SPOOL=	______,	Optional-defaults to YES
		END=	______,	Optional-defaults to NO (Specify YES if last TERMINAL statement)

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Terminal definition (continued)				
4978/4979 Terminals				
label	TERMINAL	DEVICE=	________ ,	Required (specify 4978 or 4979)
		ADDRESS=	________ ,	Required
		LINSIZE=	________ ,	Optional-defaults to 80
		TOPM=	________ ,	Optional-defaults to 0
		BOTM=	________ ,	Optional-defaults to 23
		NHIST=	________ ,	Optional-defaults to 12
		LEFTM=	________ ,	Optional-defaults to 0
		RIGHTM=	________ ,	Optional-defaults to 79
		OVFLINE=	________ ,	Optional-defaults to NO
		HDCOPY=	________ ,	Optional-defaults to $SYSPRTR
		ATTN=	________ ,	Optional-defaults to YES
		PF1=	________ ,	Optional-(2 digit hex char)-defaults to 02
		SCREEN=	________ ,	Optional-defaults to ROLL (YES/NO/STATIC)
		PART=	________ ,	Optional-defaults to partition 1
		SPOOL=	________ ,	Optional-defaults to NO
		END=	________	Optional-defaults to NO (Specify YES if last TERMINAL statement)

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Terminal definition (continued)				
4980 Terminals				
label	TERMINAL	DEVICE=**4980**,		Required
		ADDRESS=	______ ,	Required
		LINSIZE=	______ ,	Optional-defaults to 80
		TOPM=	______ ,	Optional-defaults to 0
		BOTM=	______ ,	Optional-defaults to 23
		NHIST=	______ ,	Optional-defaults to 12
		BITRATE=	______ ,	Required-defaults to 100K bps
		LEFTM=	______ ,	Optional-defaults to 0
		RIGHTM=	______ ,	Optional-defaults to 79
		OVFLINE=	______ ,	Optional-defaults to NO
		HDCOPY=	______ ,	Optional-defaults to $SYSPRTR
		ATTN=	______ ,	Optional-defaults to YES
		PF1=	______ ,	Optional-(2 digit hex char)-defaults to 02
		SCREEN=	______ ,	Optional-defaults to ROLL (YES/ROLL, STATIC or NO)
		PART=	______ ,	Optional-defaults to partition 1
		SPOOL=	______ ,	Optional-defaults to NO
		ADAPTER=**SMIO**,		Required
		PORT=	______,	Required-specify 0 or 1
		SECADDR=	______,	Required-specify 00 through FE
		END=	______	Optional-defaults to NO (Specify YES if last TERMINAL statement)

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Terminal definition (continued)				
5219/5224/5225				
label	TERMINAL	DEVICE=	______,	Required (specify 5219, 5224, or 5225)
		ADDRESS=	______,	Required
		PAGSIZE	______,	Optional-defaults to 66
		LINSIZE=	______,	Optional-defaults to 132
		CHARSET=	______,	Optional-defaults to USCA
		TOPM=	______,	Optional-defaults to 3
		BOTM=	______,	Optional-defaults to 63
		LEFTM=	______,	Optional-defaults to 0
		RIGHTM=	______,	Optional-defaults to 131
		OVFLINE=	______,	Optional-defaults to NO
		LMODE=	______,	Optional-defaults to LOCAL
		ADAPTER=**ALPA,**		Required
		SPOOL=	______,	Optional-defaults to YES
		PORT=	______,	Required-specify 0 or 1
		SECADDR=	______,	Required-specify 00 through 06
		END=	______,	Optional-defaults to NO (Specify YES if last TERMINAL statement)

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
Terminal definition (continued)				
ACCA-type Terminals				
label	TERMINAL	DEVICE=	________,	Required (specify ACCA or 3101)
		ADDRESS=	________,	Required
		MODE=	________,	Required-(3101C/3101B)
		PAGSIZE=	________,	Optional
		CODTYPE=	________,	Optional-(ASCII/EBASC)
		LINSIZE=	________,	Optional
		TOPM=	________,	Optional
		NHIST=	________,	Optional
		BOTM=	________,	Optional
		LEFTM=	________,	Optional
		RIGHTM=	________,	Optional
		OVFLINE=	________,	Optional-defaults to NO
		LINEDEL=	________,	Optional-(2 digit hex char)
		CHARDEL=	________,	Optional-(2 digit hex char)
		CRDELAY=	________,	Optional
		BITRATE=	________,	Optional-defaults to 300 bps; for MFA, defaults to 1200 bps
		RANGE=	________,	Optional-defaults to HIGH
		LMODE=	________,	Optional-(RS422/LOCAL/SWITCHED/PTTOPT)
		ADAPTER	________,	Optional-(SINGLE/TWO/FOUR SIX/EIGHT/MFA)
		COD=	________,	Optional
		CR=	________,	Optional-defaults to B0 for EBASC; defaults to 0D for ASCII
		LF=	________,	Optional-defaults to 50 for EBASC; defaults to 0A for ASCII
		ATTN=	________,	Optional-defaults to YES (NO/YES/LOCAL/NOSYS/NOGLOB)
		PF1=	________,	Optional-(2 digit hex char)-defaults to 1B61 for ASCII and D886 for EBASC
		SCREEN=	________,	Optional-(YES/NO/STATIC)
		PART=	________,	Optional-defaults to partition 1
		TIMERS=	________,	Optional
		SPOOL=	________,	Optional-defaults to NO
		END=	________	Optional-defaults to NO (Specify YES if last TERMINAL statement)

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS | **REQUIRED/OPTIONAL/DEFAULTS**

Terminal definition (continued)

TTY Terminals

label	TERMINAL	DEVICE=TTY,		Required
		ADDRESS=	______ ,	Required
		MODE=	______ ,	Optional-specify 3101C
		PAGSIZE	______ ,	Optional
		CODTYPE=ASCII,		Required
		LINSIZE=	______ ,	Optional
		TOPM=	______ ,	Optional
		BOTM=	______ ,	Optional
		LEFTM=	______ ,	Optional
		RIGHTM=	______ ,	Optional
		OVFLINE=	______ ,	Optional
		LINEDEL=	______ ,	Optional-(2 digit hex char)
		CHARDEL=	______ ,	Optional-(2 digit hex char)
		CRDELAY=	______ ,	Optional
		ECHO=	______ ,	Optional-defaults to YES
		CR=	______ ,	Optional-defaults to 0D
		LF=	______ ,	Optional-defaults to 0A
		ATTN=	______ ,	Optional-defaults to YES (NO/YES/ALL/LOCAL/NOSYS/NOGLOB)
		PF1=	______ ,	Optional-(2 digit hex char)
		SCREEN=	______ ,	Optional-(YES or ROLL/NO)
		PART=	______ ,	Optional-defaults to partition 1
		SPOOL=	______ ,	Optional-defaults to NO
		END=	______	Optional-defaults to NO (Specify YES if last TERMINAL statement)

Binary Synchronous line definition

label	BSCLINE	ADDRESS=	______ ,	Optional-defaults to 09
		TYPE=	______ ,	Optional-defaults to PT (AC/DC/PT/SM/SA/MC/MT)
		RETRIES=	______ ,	Optional-defaults to 6
		MC=	______ ,	Optional-defaults to NO
		ADAPTER=	______ ,	Optional-specify MFA
		POLL=	(______ , ______ , ______ , ______),	Optional-applies only if ADAPTER=MFA and TYPE=MT
		END=	______	Optional-defaults to NO (Specify YES if last BSCLINE statement)

IBM Series/1 Event Driven Executive

Work Sheet 2—Installation and System Generation (continued)

SYSTEM DEFINITION STATEMENTS				REQUIRED/OPTIONAL/DEFAULTS
EXIO device interface definition				
blank	EXIODEV	ADDRESS=	______,	Required
		MAXDCB=	______,	Optional-defaults to 0
		RSB=	______,	Optional-defaults to 0
		END=	______	Optional-defaults to NO (Specify YES if last EXIODEV statement)
Host communication support				
blank	HOSTCOMM	DEVICE=BSCA,		Required
		ADDRESS=	______	Required
Sensorio device definition				
blank	SENSORIO	DEVICE=	______,	Required-specify IDIO or 4982
		ADDRESS=	______,	Required
		AI=	______,	Optional
		AO=	______,	Optional
		DI	______,	Optional
		DO=	______,	Optional
		PI=	______,	Optional
		AITYPE=	______,	Optional-defaults to RELAY
		LEVEL=	______,	Optional-defaults to 1
		END=	______	Optional-defaults to NO (Specify YES if last SENSORIO statement)
System common				
	$SYSCOM	CSECT		
		QCB		
		QCB		
		ECB		
		ECB		
		ENTRY	$EDXPTCH	
	$EDXPTCH	DATA	128F'0'	System Patch Area

Work Sheet 3

Work sheet 3 helps you select the software features you need to support your configuration and the EDX-related products you are installing as part of your system.

Work sheet 3 contains the following four columns:

Column One

Use this column to indicate the modules you do not wish to include in your supervisor.

Enter an asterisk (*) in column one for each link-control statement (OPTION, PART, OVLAREA,, or INCLUDE) that is NOT required to create your supervisor. The asterisk makes the statement a comment, and the system does not include in your supervisor any object module with an asterisk. Be sure that you include the object modules required to support the processor storage characteristics and I/O devices that you defined in work sheet 2.

Supervisor Object Modules

Each entry in this column contains:

- A link-control statement (OPTION, PART, VOLUME, OVLAREA, INCLUDE,, or LINK)
- The object module name.

See Step 3 in Chapter 4, "Select Your Required Support" on page IS-39 for an explanation of each link-control statement and object module.

Notes

This column contains numbered notes. Each number refers to a note provided at the end of the $LNKCNTL data set. The notes provide a brief explanation for including a specific module.

Purpose of Module

This column describes briefly the purpose of each module.

Work Sheet 3 *(continued)*

This page intentionally left blank.

Work Sheet 3—Installation and System Generation

To include a supervisor object module in your operating system, leave Column One blank. To exclude a module, place an asterisk (*) in Column One.

Column one	Supervisor object modules	Notes	Purpose of module
	OPTION NOVERLAY	*25*	No overlay structure
Supervisor support—must be first and in partition 1			
	PART 1		
	VOLUME SX4002		Default volume for include modules
	OVLAREA OVLSTART OVLEND	*23*	User defined overlay area
	INCLUDE EDXSYS	*1*	System tables and work area
	INCLUDE ASMOBJ,EDX002	*1*	Output from user system generation
	INCLUDE EDXSVCX	*1*	Task supervisor
	INCLUDE $DEBUGNUC	*2*	Resident $DEBUG support
	INCLUDE EDXALU	*1*	EDL instruction emulator
	INCLUDE EDXSTART	*1*	Initialization & error handler
	INCLUDE SWAITM	*3*	Wait on multiple events
Timer support—must be in partition 1			
	INCLUDE EDXTIMER	*12*	4955 timer (7840) support
	INCLUDE EDXTIMR2	*12*	4952/4954/4956 timer support
EXIO support—must be in partition 1			
	INCLUDE IOSEXIO	*3*	EXIO device control support
	INCLUDE EXIOTRC	*3*	EXIO trace option
Error logging—must be in partition 1			
	INCLUDE SYSLOG	*3*	I/O error logging
	INCLUDE CIRCBUFF	*16*	Program/machine check logging
Optional function support—must be in partition 1			
	INCLUDE RLOADER	*17*	Relocating program loader
	INCLUDE STORMGR	*3*	Unmapped storage manager support
	INCLUDE IAMQCB	*20*	IAMQCB needed for IAM support
	INCLUDE PWRAM80	*26*	4980 Power on RAM
Host communications support—must be in partition 1			
	INCLUDE TPCOM	*14*	Host communication support
Translation tables—must be in partition 1			
	INCLUDE TRASCII	*10*	1310,2095/2096,7850 ACCA/TTY translation
	INCLUDE TREBASC	*10*	1610,2091/2092 ACCA translation
	INCLUDE TREBCD	*11*	2741,PROC EBDC translation
	INCLUDE TRCRSP	*11*	2741 correspondence translation
Disk(ette) and tape support—must be grouped together in any partition			
	PART 1		
	INCLUDE DISKIO	*3*	Basic disk(ette) support
	INCLUDE D49624	*3*	4962/4964 disk(ette) support
	INCLUDE D4963A	*3*	4963/4967/DDSK-30 subsystem support
	INCLUDE D4966A	*3*	4965/4966 disk(ette) support
	INCLUDE D1024	*3,21*	1024 bytes/sector IO support
	INCLUDE D4969A	*3*	Basic tape support
Terminal—must be grouped together in any partition			
	PART 1		
	INCLUDE EDXTIO	*4*	Basic terminal support
	INCLUDE MINMSG	*5*	Message ID only support
	INCLUDE FULLMSG	*5*	Message data set support
	INCLUDE IOS3101	*7*	ACCA 3101B support
	INCLUDE IOS4979	*3*	4978/4979/4980 display support

Figure 25. Work Sheet 3

Work Sheet 3—Installation and System Generation (cont.)

Column one	Supervisor object modules	Notes	Purpose of module
	INCLUDE IOS4974	*3*	4973/4974/4975/5219/5224/5225 printer support
	INCLUDE IOSACCA	*4*	ACCA device handler
	INCLUDE ACCATRC	*3*	ACCA trace option
	INCLUDE IOSTERM	*A,6*	ACCA/TTY/2741/4013 support
	INCLUDE IOSTTY	*3*	ASR 33/35/3101/3101C TTY support
	INCLUDE IOS2741	*3*	2741 terminal support
	INCLUDE IOS4013	*3*	Digital I/O terminal support
	INCLUDE IOSGPIB	*3*	GPIB support
	INCLUDE IOSS1S1	*3*	Series/1 - Series/1 support
	INCLUDE IOSVIRT	*3,9*	Virtual terminal support
	INCLUDE IOSPOOL	*3*	Spooling support
	INCLUDE EBFLCVT	*18*	EBCDIC/floating point conversion
Floating point—may be in any partition			
	PART 1		
	INCLUDE EDXFLOAT	*3*	Floating point arithmetic
Queue I/O—may be in any partition			
	PART 1		
	INCLUDE QUEUEIO	*19*	Queue processing support
Bisync communications—may be in any partition			
	PART 1		
	INCLUDE BSCAM	*13*	Bisync communication support
	INCLUDE BSCX21	*27*	Bisync communication support for X.21
Sensor input/output—must be grouped together in any partition			
	PART 1		
	INCLUDE SBCOM	*15*	Basic sensor I/O support
	INCLUDE SBAI	*3*	Analog input support
	INCLUDE SBAO	*3*	Analog output support
	INCLUDE SBDIDO	*3*	Digital input/output support
	INCLUDE SBPI	*3*	Process interrupt support
System initialization—must be in partition 1			
	INCLUDE $PROG1	*22*	User module included in nucleus gen
	INCLUDE IO1024	*21*	1024 IPL support
System support -- initialization—must be in partition 1			
	INCLUDE EDXINIT	*24*	Supervisor initialization
	INCLUDE RW4963ID	*3*	4963 fixed head refresh support
	INCLUDE INITMFA	*3*	MFA attachment initialization
	INCLUDE INITADAP	*3*	ALPA and SMIO initialization
	INCLUDE INIT4978	*3*	4978/4979 terminal initialization
	INCLUDE INIT4980	*3*	4980 display initialization

Work Sheet 3—Installation and System Generation (cont.)

Column one	Supervisor object modules	Notes	Purpose of module
Insert user initialization modules here			

IBM Series/1 Event Driven Executive

Work Sheet 3—Installation and System Generation (cont.)

Column one	Supervisor object modules	Notes	Purpose of module
Supervisor code being moved out of partition 1 must be moved to here			
Location of supervisor			
LINK $EDXNUCT,EDX002 REPLACE END			

Work Sheet 4

Work sheet 4 helps you estimate the size of those portions of the supervisor that are relocated to partitions other than partition one and the amount of storage required by programs to execute within a partition.

Work sheet 4 is made up of two parts: A and B.

Part A Lists the approximate sizes (in bytes) of the supervisor object modules that make up the supervisor. This part is used to estimate what portion of each partition the supervisor will use.

Part B Lists the approximate amount of storage required by the supervisor to support program products and utility programs. In addition, application programs require a certain amount of storage within a partition in order to execute. The algorithms to estimate these storage requirements are provided.

PART A

Supervisor Object Modules

Each supervisor object module included in your supervisor requires processor storage. Some of these modules MUST remain in partition 1; others you can locate outside of partition 1.

Under "Partition One:" on page IS-316 is a list of object modules that must remain in partition 1 and their approximate sizes (in bytes). Under "Partition Any:" on page IS-317 are the names of object modules that can be located outside of partition 1, the names of the root segment that is left in partition 1 if a module is located outside of partition 1, and their approximate sizes (in bytes).

Work Sheet 4 *(continued)*

Partition One: The following supervisor object modules must be located in partition 1. Compare this list against work sheet 3 and select the object modules that you are including in your system. Cross out the modules you are not including, along with their corresponding sizes.

Total the SIZE column and round the total upward to the next multiple of 256. The resulting figure represents the amount of storage required by the supervisor to support these modules.

MODULE NAME	SIZE (IN BYTES)
EDXSYS	1412
EDXSVCX	2758
$DBUGNUC	646
EDXALU	2360
EDXSTART	2190
SWAITM	852
EDXTIMER	1326
EDXTIMR2	1320
IOSEXIO	1078
EXIOTRC	774
SYSLOG	236
CIRCBUFF	252
RLOADER	5922
STORMGR	1352
IAMQCB	26
PWRAM80	364
TPCOM	2020
TRASCII	514
TREBASC	514
TREBCD	514
TRCRSP	514
IO1024	2896
EDXINIT	10
RW4963ID	526
INITMFA	1398
INITADAP	2478
INIT4978	2338
INIT4980	2550
User initialization modules	

Figure 26. Supervisor Modules Required in Partition 1

Work Sheet 4 *(continued)*

The linkage editor, $XPSLINK, automatically includes the following initialization modules in your supervisor. They are included as overlay segments if you default to overlay structure or as resident programs if you include the OPTION NOVERLAY statement. If $XPSLINK includes them as resident programs, the initialization modules required for your system are loaded simultaneously in supervisor storage. As a result, they increase the amount of storage required by the supervisor in partition 1.

To determine how much additional storage you require, select the initialization modules you require to support the I/O devices attached to your Series/1. Cross out the modules you do not need, along with their corresponding sizes. Total the SIZE column; add the resulting figure to the total of the SIZE column in Figure 26 on page IS-316 .

MODULE NAME	SIZE (IN BYTES)
BSCINIT	578
CLOKINIT	188
DISKINIT	1634
DSKINIT1	1760
DSKINIT2	1194
EXIOINIT	66
INIT4013	182
LOADINIT	1568
SBIOINIT	180
S1S1INIT	274
STORINIT	422
TAPEINIT	1704
TERMINIT	1714
TIMRINIT	356
TPINIT	364

Partition Any: The following supervisor object modules can be located in partition 2 through 8 (depending upon your processor). Compare this list against work sheet 3 and select the object modules that you are including in your system. Cross out the modules you are not including, along with their corresponding sizes.

If you are generating a single-partition supervisor and leave these modules located in partition 1, cross out the modules you do not need, along with their corresponding sizes. Total the SIZE column and add the resulting figure to the total of the SIZE column in Figure 26 on page IS-316.

Work Sheet 4 *(continued)*

If you are generating a multipartition supervisor and move any of these modules into partitions 2 through 8, you can estimate the amount of storage required within a specific partition by the size shown in the SIZE column. As indicated by the horizontal rule, some of these modules MUST be grouped together. If you move a group of modules outside of partition 1, add up the number of bytes for the entire group (excluding those not included).

In addition, $XPSLINK includes a 'root' segment in partition 1 that corresponds to each supervisor object module being located outside of partition 1. Each root segment requires an amount of storage within the supervisor in partition 1. For each object module that you locate outside of partition 1, add the amount shown in the ROOT SEGMENT column to the total in the SIZE column in Figure 26 on page IS-316.

MODULE NAME	SIZE (IN BYTES)	ROOT SEGMENT
DISKIO	1478	744
D1024	312	158
D49624	1546	22
D4963A	632	18
D4966A	994	644
D4969A	3994	1280
EDXTIO	4818	990
MINMSG	244	0
FULLMSG	1362	0
IOS3101	2196	0
IOS4974	1132	86
IOS4975A	734	134
IOS4979	2684	292
IOSTERM	676	0
IOSACCA	1296	392
ACCATRC	714	368
IOSTTY	172	492
IOS2741	878	678
IOS4013	282	310
IOSGPIB	676	134
IOSS1S1	986	344
IOSVIRT	708	0
IOSPOOL	2050	206
EBFLCVT	1576	0
EDXFLOAT	470	56
QUEUEIO	312	14
BSCAM	4230	98
BSCX21	456	100
SBCOM	140	12
SBAI	640	62
SBAO	76	14
SBDIDO	826	188
SBPI	2	166

Figure 27. Supervisor Modules Not Required in Partition One

Work Sheet 4 *(continued)*

PART B

Program Products

The following is a list of products that are not part of the supervisor but that use storage in partition one if you include them in your system. Read down the list and cross out the products you are not including in your system. Total the RESIDENT column and round the total upward to the next multiple of 256. The resulting figure represents the amount of storage required by the supervisor to support these products.

Support	Resident
Data set open (DSOPEN)	1550
5230 Data Collection	
Interactive	
1 to 31 entry units	16384
31 to 63 entry units	18432
System/370 Channel Attach	
$CAPGM (in a partition)	7500
link module (in a program)	3000
$CHANUT1 (in a partition)	5400
+900 per device	( )
Multiple Terminal Manager	13312
+522 if indexed support	( )
+552 if 3101 basic	( )
+terminal servers	
(rounded to next 256-byte block)	
834*no. of 4978/4979	( )
1452*no. of 3101 in block mode	( )
876*no. of ASR 33/35	( )
Support of 1024 bytes per 2200 sector diskette	
Indexed Access Method	(3400)
Version 1	
+625*no. of data sets	( )
+250*no. of application programs	( )

The Indexed Access Method Version 2 is supplied as four different packages. You install the package you require based on storage function and requirements. Storage requirements vary from a minimum of 15K bytes to a maximum of 27K bytes. Control blocks, buffers, and work areas require additional storage. Refer to the Indexed Access Method Version 2 Programming Directory for further information.

Work Sheet 4 *(continued)*

Utility Programs

The following is a list of the approximate sizes of each EDX system utility. In planning your system, you need to ensure that the partition in which you plan to use a specific utility is large enough. To ensure that there is enough storage in a partition to execute a utility, you need to take into account the size of the largest utility you intend to use and consider its size along with program products and applications programs you execute within a specific partition.

The approximate size in bytes (rounded up to the next 256-byte increment) required by each utility is as follows:

UTILITY	SIZE	
$BSCTRCE	2560	
$BSCUT1	7680	
$BSCUT2	24046	
$COMPRES	12800	
$COPY	13824	
$COPYUT1	16128	
$DASDI	25600	
$DEBUG	12032	
$DICOMP	15872	
$DIINTR	13056	
$DIRECT	12544	
$DISKUT1	20224	
$DISKUT2	18432	(+1280 if printing error log)
$DISKUT3	11008	
$DIUTIL	13568	
$DUMP	10764	
$EDIT1	13824	
$EDIT1N	17920	
$EDXASM	19712	(+5632 if assembling TERMINAL statements)
$EDXLINK	38912	
$EDXLIST	13056	
$FONT	20736	
$FSEDIT	22528	
$GPIBUT1	15104	
$HCFUT1	3328	
$HXUT1	23808	
$IAMUT1	19712	(used for Indexed Access Method Versions 1 and 2)
$IMAGE	12288	
$INITDSK	30208	
$IOTEST	14336	
$JOBQUT	7680	
$JOBUTIL	1280	
$LOG	7680	
$MOVEVOL	5120	
$MSGUT1	28416	

Work Sheet 4 *(continued)*

UTILITY	SIZE
$PDS	2816
$PFMAP	768
$PREFIND	8704
$PRT2780	3072
$PRT3780	3328
$RJE2780	14080
$RJE3780	14848
$RMU	3584
$SPLUT1	5120
$S1S1UT1	11520
$STGUT1	7424
$SUBMIT	8960
$TAPEUT1	29110
$TERMUT1	11520
$TERMUT2	12288
$TERMUT3	1280
$TRACEIO	14848
$TRAP	7424
$UPDATE	13056
$UPDATEH	9728
$XPSLINK	23702

Application Programs

Programs Written in Event Driven Language

Estimate the storage required for an EDL program by totaling the following:

```
Program Name: ______________   Date:________   Filled Out By: ___________________
_________________________________________________________________________________

1.  Number of source statements * 10                     =  (       )
2.  Size of large tables and buffers                     =  (       )
3.  Graphics subroutines                                 =  (       )
4.  Formatting subroutines                               =  (       )
5.  Formatting instructions                              =  (       )
6.  Formatted screen image subroutines                   =  (       )
7.  Programs using assembler language code               =  (       )
8.  Extra task control block (for ATTNLIST)              =  (       )
9.  DSOPEN subroutine                                    =  (   1550)
10. SETEOD subroutine                                    =  (    280)
11. $DISKUT3 data management utility                     =  (   5200)
12. $LOADER transient program loader                     =  (   5704)
13. $UPCASE                                              =  (    150)
14. $4975                                                =  (    300)
                                                             _______
TOTAL                                                    =  (       )
```

The following notes supply additional information on how to determine these storage items:

Work Sheet 4 *(continued)*

Notes:

1. The number of source statements includes operation code and instruction parameters plus incidental tables and buffers.

2. The number of words coded for tables and buffers.

3. Graphics instructions cause subroutines to be added to your program. Add the following for the first occurrence of each instruction:

```
MODULE                   SIZE (BYTES)
CONCAT                     304
GIN                        176
PLOTCB                      16
PLOTGIN                    204  (+GIN if not already used)
SCREEN                     492
XYPLOT                     368  (+SCREEN and CONCAT if
                                not already used)
YTPLOT                     368  (+SCREEN if not already used)

TOTAL                    (_____)
```

4. If you use data formatting instructions, the compiler includes FORMATTING SUBROUTINES in your program. Add the number indicated for each first occurrence of the following specifications included in FORMAT statements referenced by instructions.

Instruction	Specification	Bytes
GETEDIT,PUTEDIT or FORMAT	any	1014
GETEDIT	alphameric	202
GETEDIT	float.pt. F or E	96
GETEDIT	integer	90
PUTEDIT	alphameric	36
PUTEDIT	float.pt. F	82
PUTEDIT	float.pt. E	82
PUTEDIT	integer	76
GETEDIT or PUTEDIT	any parenthetical expression	22

5. If you use data formatting instructions, the compiler inserts data areas into your program. Add "B" bytes for each occurrence of the following instructions:

```
GETEDIT
  B=16+(4*V)+A
  where V=the number of variables in list
        A=6 if ACTION=IO, else A=0
```

Work Sheet 4 *(continued)*

```
PUTEDIT
  B=16+(4*V)+A
  where V=number of variables in list
        A=4 if ACTION=IO, else A=0
```

```
FORMAT
  B=24+(4*L)
  where L=number of elements in FORMAT list
```

6. When you include the formatted screen image subroutines, the size of your program increases by the amount shown for each subroutine included:

MODULE	SIZE (BYTES)
$IMOPEN	2600
$PACK	112
$UNPACK	64
$IMDTYPE	116
$IMGEN (includes $IMDEFN, $IMPROT, $IMDATA)	3626
$IMGEN31	2060
$IMGEN49	1194
$IMGEN3X	3310
TOTAL	(______)

7. Programs containing assembler language code require you to include one or more of the following subroutines. Add the number of bytes shown for EACH OCCURRENCE of the following instructions:

MODULE	SIZE (BYTES)	X	OCCURRENCE		
$$RETURN (entry point RETURN)	56	X	\_\_\_\_\_\_	=	\_\_\_\_\_\_
$$SVC (entry point SVC)	20	X	______	=	______
$EDXATSR (entry points) - SETBUSY, SUPEXIT	58	X	______	=	______
TOTAL					(______)

8. When you code a local or a global (or both) ATTNLIST, it generates an extra 128-byte TCB.

9. When you use DSOPEN in a program, the size of the program increases by 1500 bytes.

10. When you use SETEOD in a program, the size of the program increases by 280 bytes.

11. When you use $DISKUT3 in a program, it requires an additional 4600 bytes of storage.

Work Sheet 4 *(continued)*

12. The transient program loader ($LOADER) requires an area of 4900 bytes which is overlaid when the program is loaded.

13. When you use $UPCASE in a program, it requires an additional 150 bytes of storage.

14. When you use $4975 in a program, it requires an additional 300 bytes of storage.

Application Program Storage Estimating

Programs Written in COBOL

Estimate the storage required for a COBOL program by adding the following:

1. 11,000 bytes.
2. Number of PROCEDURE DIVISION statements times 12.
3. Space used by items in DATA DIVISION as indicated on compiler listing if you specified MAP.
4. Block size of each file times 2.

Programs Written in PL/I

Estimate the storage required for a PL/I program by adding the following:

1. 20,000 bytes.
2. Number of statements times 52.

Programs Written in FORTRAN

Estimate the storage required for a FORTRAN program by adding the following:

1. 11,000 bytes.

2. Sum of the number of bytes shown in TOTAL PROGRAM REQUIREMENTS at the end of the compiler listing for each routine.

3. Space used by EDL driver program.

Programs Written in Pascal

Estimate the storage required for a Pascal program by adding the following:

1. 15,000 bytes.

2. The number of bytes shown in TOTAL PROGRAM REQUIREMENTS at the end of the compiler listing for each module.

Glossary of Terms and Abbreviations

This glossary defines terms and abbreviations used in the Series/1 Event Driven Executive software publications. All software and hardware terms pertain to EDX. This glossary also serves as a supplement to the *IBM Data Processing Glossary*, GC20-1699.

$SYSLOGA, $SYSLOGB. The name of the alternate system logging device. This device is optional but, if defined, should be a terminal with keyboard capability, not just a printer.

$SYSLOG. The name of the system logging device or operator station; must be defined for every system. It should be a terminal with keyboard capability, not just a printer.

$SYSPRTR. The name of the system printer.

abend. Abnormal end-of-task. Termination of a task prior to its completion because of an error condition that cannot be resolved by recovery facilities while the task is executing.

ACCA. See asynchronous communications control adapter.

address key. Identifies a set of Series/1 segmentation registers and represents an address space. It is one less than the partition number.

address space. The logical storage identified by an address key. An address space is the storage for a partition.

application program manager. The component of the Multiple Terminal Manager that provides the program management facilities required to process user requests. It controls the contents of a program area and the execution of programs within the area.

application program stub. A collection of subroutines that are appended to a program by the linkage editor to provide the link from the application program to the Multiple Terminal Manager facilities.

asynchronous communications control adapter. An ASCII terminal attached via #1610, #2091 with #2092, or #2095 with #2096 adapters.

attention key. The key on the display terminal keyboard that, if pressed, tells the operating system that you are entering a command.

attention list. A series of pairs of 1 to 8 byte EBCDIC strings and addresses pointing to EDL instructions. When the attention key is pressed on the terminal, the operator can enter one of the strings to cause the associated EDL instructions to be executed.

backup. A copy of data to be used in the event the original data is lost or damaged.

base record slots. Space in an indexed file that is reserved for based records to be placed.

base records. Records are placed into an indexed file while in load mode or inserted in process mode with a new high key.

basic exchange format. A standard format for exchanging data on diskettes between systems or devices.

binary synchronous device data block (BSCDDB). A control block that provides the information to control one Series/1 Binary Synchronous Adapter. It determines the line characteristics and provides dedicated storage for that line.

Glossary of Terms and Abbreviations

block. (1) See data block or index block. (2) In the Indexed Method, the unit of space used by the access method to contain indexes and data.

block mode. The transmission mode in which the 3101 Display Station transmits a data data stream, which has been edited and stored, when the SEND key is pressed.

BSCAM. See binary synchronous communications access method.

binary synchronous communications access method. A form of binary synchronous I/O control used by the Series/1 to perform data communications between local or remote stations.

BSCDDB. See binary synchronous device data block.

buffer. An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. See input buffer and output buffer.

bypass label processing. Access of a tape without any label processing support.

CCB. See terminal control block.

central buffer. The buffer used by the Indexed Access Method for all transfers of information between main storage and indexed files.

character image. An alphabetic, numeric, or special character defined for an IBM 4978 Display Station. Each character image is defined by a dot matrix that is coded into eight bytes.

character image table. An area containing the 256 character images that can be defined for an IBM 4978 Display Station. Each character image is coded into eight bytes, the entire table of codes requiring 2048 bytes of storage.

character mode. The transmission mode in which the 3101 Display Station immediately sends a character when a keyboard key is pressed.

cluster. In an indexed file, a group of data blocks that is pointed to from the same primary-level index block, and includes the primary-level index block. The data records and blocks contained in a cluster are logically contiguous, but are not necessarily physically contiguous.

COD (change of direction). A character used with ACCA terminal to indicate a reverse in the direction of data movement.

cold start. Starting the spool facility by erasing any spooled jobs remaining in the spool data set from any previous spool session.

command. A character string from a source external to the system that represents a request for action by the system.

common area. A user-defined data area that is mapped into the partitions specified on the SYSTEM definition statement. It can be used to contain control blocks or data that will be accessed by more than one program.

completion code. An indicator that reflects the status of the execution of a program. The completion code is displayed or printed on the program's output device.

constant. A value or address that remains unchanged thoughout program execution.

controller. A device that has the capability of configuring the GPIB bus by designating which devices are active, which devices are listeners, and which device is the talker. In Series/1 GPIB implementation, the Series/1 is always the controller.

conversion. See update.

control station. In BSCAM communications, the station that supervises a multipoint connection, and performs polling and selection of its tributary stations. The status of control station is assigned to a BSC line during system generation.

cross-partition service. A function that accesses data in two partitions.

cross-partition supervisor. A supervisor in which one or more supervisor modules reside outside of partition 1 (address space 0).

data block. In an indexed file, an area that contains control information and data records. These blocks are a multiple of 256 bytes.

data record. In an indexed file, the records containing customer data.

data set. A group of records within a volume pointed to by a directory member entry in the directory for the volume.

data set control block (DSCB). A control block that provides the information required to access a data set, volume or directory using READ and WRITE.

data set shut down. An indexed data set that has been marked (in main storage only) as unusable due to an error.

DCE. See directory control entry.

device data block (DDB). A control block that describes a disk or diskette volume.

direct access. (1) The access method used to READ or WRITE records on a disk or diskette device by specifying their location relative the beginning of the data set or volume. (2) In the Indexed Access Method, locating any record via its key without respect to the previous operation. (3) A condition in terminal I/O where a READTEXT or a PRINTEXT is directed to a buffer which was previously enqueued upon by an IOCB.

directory. (1) A series of contiguous records in a volume that describe the contents in terms of allocated data sets and free space. (2) A series of contiguous records on a device that describe the contents in terms of allocated volumes and free space. (3) For the Indexed Access Method Version 2, a data set that defines the relationship between primary and secondary indexed files (secondary index support).

directory control entry (DCE). The first 32 bytes of the first record of a directory in which a description of the directory is stored.

directory member entry (DME). A 32-byte directory entry describing an allocated data set or volume.

display station. An IBM 4978, 4979, or 3101 display terminal or similar terminal with a keyboard and a video display.

DME. See directory member entry.

DSCB. See data set control block.

dynamic storage. An increment of storage that is appended to a program when it is loaded.

end-of-data indicator. A code that signals that the last record of a data set has been read or written. End-of-data is determined by an end-of-data pointer in the DME or by the physical end of the data set.

ECB. See event control block.

EDL. See Event Driven Language.

emulator. The portion of the Event Driven Executive supervisor that interprets EDL instructions and performs the function specified by each EDL statement.

end-of-tape (EOT). A reflective marker placed near the end of a tape and sensed during output. The marker signals that the tape is nearly full.

enter key. The key on the display terminal keyboard that, if pressed, tells the operating system to read the information you entered.

event control block (ECB). A control block used to record the status (occurred or not occurred) of an event; often used to synchronize the execution of tasks. ECBs are used in conjunction with the WAIT and POST instructions.

Event Driven Language (EDL). The language for input to the Event Driven Executive compiler ($EDXASM), or the Macro and Host assemblers in conjunction with the Event Driven Executive macro libraries. The output is interpreted by the Event Driven Executive emulator.

EXIO (execute input or output). An EDL facility that provides user controlled access to Series/1 input/output devices.

external label. A label attached to the outside of a tape that identifies the tape visually. It usually contains items of identification such as file name and number, creation data, number of volumes, department number, and so on.

external name (EXTRN). The 1- to 8-character symbolic EBCDIC name for an entry point or data field that is not defined within the module that references the name.

FCA. See file control area.

FCB. See file control block.

file. A set of related records treated as a logical unit. Although file is often used interchangeably with data set, it usually refers to an indexed or a sequential data set.

file control area (FCA). A Multiple Terminal Manager data area that describes a file access request.

file control block (FCB). The first block of an indexed file. It contains descriptive information about the data contained in the file.

file control block extension. The second block of an indexed file. It contains the file definition parameters used to define the file.

file manager. A collection of subroutines contained within the program manager of the Multiple Terminal Manager that provides common support for all disk data transfer operations as needed for transaction-oriented application programs. It supports indexed and direct files under the control of a single callable function.

floating point. A positive or negative number that can have a decimal point.

formatted screen image. A collection of display elements or display groups (such as operator prompts and field input names and areas) that are presented together at one time on a display device.

free pool. In an indexed data set, a group of blocks that can be used for either data blocks or index blocks. These differ from other free blocks in that these are not initially assigned to specific logical positions in the file.

free space. In an indexed file, records blocks that do not currently contain data, and are available for use.

free space entry (FSE). An 8-byte directory entry defining an area of free space within a volume or a device.

FSE. See free space entry.

general purpose interface bus. The IEEE Standard 488-1975 that allows various interconnected devices to be attached to the GPIB adapter (RPQ D02118).

Glossary of Terms and Abbreviations

GPIB. See general purpose interface bus.

group. A unit of 100 records in the spool data set allocated to a spool job.

H exchange format. A standard format for exchanging data on diskettes between systems or devices.

host assembler. The assembler licensed program that executes in a 370 (host) system and produces object output for the Series/1. The source input to the host assembler is coded in Event Driven Language or Series/1 assembler language. The host assembler refers to the System/370 Program Preparation Facility (5798-NNQ).

host system. Any system whose resources are used to perform services such as program preparation for a Series/1. It can be connected to a Series/1 by a communications link.

IACB. See indexed access control block.

IAR. See instruction address register.

ICB. See indexed access control block.

IIB. See interrupt information byte.

image store. The area in a 4978 that contains the character image table.

immediate data. A self-defining term used as the operand of an instruction. It consists of numbers, messages or values which are processed directly by the computer and which do not serve as addresses or pointers to other data in storage.

index. In an indexed file, an ordered collection of pairs of keys and pointers, used to sequence and locate records.

index block. In an indexed file, an area that contains control information and index entries. These blocks are a multiple of 256 bytes.

indexed access control block (IACB/ICB). The control block that relates an application program to an indexed file.

indexed access method. An access method for direct or sequential processing of fixed-length records by use of a record's key.

indexed data set. Synonym for indexed file.

indexed file. A file specifically created, formatted and used by the Indexed Access Method. An indexed file is sometimes called an indexed data set.

index entry. In an indexed file, a key-pointer pair, where the pointer is used to locate a lower-level index block or a data block.

index register (#1, #2). Two words defined in EDL and contained in the task control block for each task. They are used to contain data or for address computation.

input buffer. (1) See buffer. (2) In the Multiple Terminal Manager, an area for terminal input and output.

input output control block (IOCB). A control block containing information about a terminal such as the symbolic name, size and shape of screen, the size of the forms in a printer, or an optional reference to a user provided buffer.

instruction address register (IAR). The pointer that identifies the machine instruction currently being executed. The Series/1 maintains a hardware IAR to determine the Series/1 assembler instruction being executed. It is located in the level status block (LSB).

integer. A positive or negative number that has no decimal point.

interactive. The mode in which a program conducts a continuous dialogue between the user and the system.

internal label. An area on tape used to record identifying information (similar to the identifying information placed on an external label). Internal labels are checked by the system to ensure that the correct volume is mounted.

interrupt information byte (IIB). In the Multiple Terminal Manager, a word containing the status of a previous input/output request to or from a terminal.

invoke. To load and activate a program, utility, procedure, or subroutine into storage so it can run.

job. A collection of related program execution requests presented in the form of job control statements, identified to the jobstream processor by a JOB statement.

job control statement. A statement in a job that specifies requests for program execution, program parameters, data set definitions, sequence of execution, and, in general, describes the environment required to execute the program.

job stream processor. The job processing facility that reads job control statements and processes the requests made by these statements. The Event Driven Executive job stream processor is $JOBUTIL.

jumper. (1) A wire or pair of wires which are used for the arbitrary connection between two circuits or pins in an attachment card. (2) To connect wire(s) to an attachment card or to connect two circuits.

key. In the Indexed Access Method, one or more consecutive characters used to identify a record and establish its order with respect to other records. See also key field.

key field. A field, located in the same position in each record of an indexed file, whose content is used for the key of a record.

level status block (LSB). A Series/1 hardware data area that contains processor status. This area is eleven words in length.

library. A set of contiguous records within a volume. It contains a directory, data sets and/or available space.

line. A string of characters accepted by the system as a single input from a terminal; for example, all characters entered before the carriage return on the teletypewriter or the ENTER key on the display station is pressed.

link edit. The process of resolving external symbols in one or more object modules. A link edit is performed with $EDXLINK whose output is a loadable program.

listener. A controller or active device on a GPIB bus that is configured to accept information from the bus.

load mode. In the Indexed Access Method, the mode in which records are loaded into base record slots in an indexed file.

load module. A single module having cross references resolved and prepared for loading into storage for execution. The module is the output of the $UPDATE or $UPDATEH utility.

load point. (1) Address in the partition where a program is loaded. (2) A reflective marker placed near the beginning of a tape to indicate where the first record is written.

lock. In the Indexed Access Method, a method of indicating that a record or block is in use and is not available for another request.

logical screen. A screen defined by margin settings, such as the TOPM, BOTM, LEFTM and RIGHTM parameters of the TERMINAL or IOCB statement.

LSB. See level status block.

mapped storage. The processor storage that you defined on the SYSTEM statement during system generation.

member. A term used to identify a named portion of a partitioned data set (PDS). Sometimes member is also used as a synonym for a data set. See data set.

menu. A formatted screen image containing a list of options. The user selects an option to invoke a program.

menu-driven. The mode of processing in which input consists of the responses to prompting from an option menu.

message. In data communications, the data sent from one station to another in a single transmission. Stations communication with a series of exchanged messages.

multifile volume. A unit of recording media, such as tape reel or disk pack, that contains more than one data file.

multiple terminal manager. An Event Driven Executive licensed program that provides support for transaction-oriented applications on a Series/1. It provides the capability to define transactions and manage the programs that support those transactions. It also manages multiple terminals as needed to support these transactions.

multivolume file. A data file that, due to its size, requires more than one unit of recording media (such as tape reel or disk pack) to contain the entire file.

new high key. A key higher than any other key in an indexed file.

nonlabeled tapes. Tapes that do not contain identifying labels (as in standard labeled tapes) and contain only files separated by tapemarks.

null character. A user-defined character used to define the unprotected fields of a formatted screen.

option selection menu. A full screen display used by the Session Manager to point to other menus or system functions, one of which is to be selected by the operator. (See primary option menu and secondary option menu.)

output buffer. (1) See buffer. (2) In the Multiple Terminal Manager, an area used for screen output and to pass data to subsequent transaction programs.

overlay. The technique of reusing a single storage area allocated to a program during execution. The storage area can be reused by loading it with overlay programs that have been specified in the PROGRAM statement of the program or by calling overlay segments that have been specified in the OVERLAY statement of $EDXLINK.

overlay area. A storage area within a program reserved for overlay programs specified in the PROGRAM statement or overlay segments specified in the OVERLAY statement in $EDXLINK.

overlay program. A program in which certain control sections can use the same storage location at different times during execution. An overlay program can execute concurrently as an asynchronous task with other programs and is specified in the EDL PROGRAM statement in the main program.

overlay segment. A self-contained portion of a program that is called and sequentially executes as a synchronous task. The entire program that calls the overlay segment need not be maintained in storage while the overlay segment is executing. An overlay segment is specified in the OVERLAY statement of $EDXLINK or $XPSLINK (for initialization modules).

overlay segment area. A storage area within a program or supervisor reserved for overlay segments. An overlay segment area is specified with the OVLAREA statement of $EDXLINK.

Glossary of Terms and Abbreviations

parameter selection menu. A full screen display used by the Session Manager to indicate the parameters to be passed to a program.

partition. A contiguous fixed-sized area of storage. Each partition is a separate address space.

performance volume. A volume whose name is specified on the DISK definition statement so that its address is found during IPL, increasing system performance when a program accesses the volume.

physical timer. Synonym for timer (hardware).

polling. In data communications, the process by which a multipoint control station asks a tributary if it can receive messages.

precision. The number of words in storage needed to contain a value in an operation.

prefind. To locate the data sets or overlay programs to be used by a program and to store the necessary information so that the time required to load the prefound items is reduced.

primary file. An indexed file containing the data records and primary index.

primary file entry. For the Indexed Access Method Version 2, an entry in the directory describing a primary file.

primary index. The index portion of a primary file. This is used to access data records when the primary key is specified.

primary key. In an indexed file, the key used to uniquely identify a data record.

primary-level index block. In an indexed file, the lowest level index block. It contains the relative block numbers (RBNs) and high keys of several data blocks. See cluster.

primary menu. The program selection screen displayed by the Multiple Terminal Manager.

primary option menu. The first full screen display provided by the Session Manager.

primary station. In a Series/1 to Series/1 attachment, the processor that control communication between the two computers. Contrast with secondary station.

primary task. The first task executed by the supervisor when a program is loaded into storage. It is identified by the PROGRAM statement.

priority. A combination of hardware interrupt level priority and a software ranking within a level. Both primary and secondary tasks will execute asynchronously within the system according to the priority assigned to them.

process mode. In the Indexed Access Method, the mode in which records can be retrieved, updated, inserted or deleted.

processor status word (PSW). A 16-bit register used to (1) record error or exception conditions that may prevent further processing and (2) hold certain flags that aid in error recovery.

program. A disk- or diskette-resident collection of one or more tasks defined by a PROGRAM statement; the unit that is loaded into storage. (See primary task and secondary task.)

program header. The control block found at the beginning of a program that identifies the primary task, data sets, storage requirements and other resources required by a program.

program/storage manager. A component of the Multiple Terminal Manager that controls the execution and flow of application programs within a single program area and contains the support needed to allow multiple operations and sharing of the program area.

protected field. A field in which the operator cannot use the keyboard to enter, modify, or erase data.

PSW. See processor status word.

QCB. See queue control block.

QD. See queue descriptor.

QE. See queue element.

queue control block (QCB). A data area used to serialize access to resources that cannot be shared. See serially reusable resource.

queue descriptor (QD). A control block describing a queue built by the DEFINEQ instruction.

queue element (QE). An entry in the queue defined by the queue descriptor.

quiesce. To bring a device or a system to a halt by rejection of new requests for work.

quiesce protocol. A method of communication in one direction at a time. When sending node wants to receive, it releases the other node from its quiesced state.

record. (1) The smallest unit of direct access storage that can be accessed by an application program on a disk or diskette using READ and WRITE. Records are 256 bytes in length. (2) In the Indexed Access Method, the logical unit that is transferred between $IAM and the user's buffer. The length of the buffer is defined by the user. (3) In BSCAM communications, the portions of data transmitted in a message. Record length (and, therefore, message length) can be variable.

recovery. The use of backup data to re-create data that has been lost or damaged.

reflective marker. A small adhesive marker attached to the reverse (nonrecording) surface of a reel of magnetic tape. Normally, two reflective markers are used on each reel of tape. One indicates the beginning of the recording area on the tape (load point), and the other indicates the proximity to the end of the recording area (EOT) on the reel.

relative block address (RBA). The location of a block of data on a 4967 disk relative to the start of the device.

relative record number. An integer value identifying the position of a record in a data set relative to the beginning of the data set. The first record of a data set is record one, the second is record two, the third is record three.

relocation dictionary (RLD). The part of an object module or load module that is used to identify address and name constants that must be adjusted by the relocating loader.

remote management utility control block (RCB). A control block that provides information for the execution of remote management utility functions.

reorganize. The process of copying the data in an indexed file to another indexed file in a manner that rearranges the data for more optimum processing and free space distribution.

restart. Starting the spool facility w the spool data set contains jobs from a previous session. The jobs in the spool data set can be either deleted or printed when the spool facility is restarted.

return code. An indicator that reflects the results of the execution of an instruction or subroutine. The return code is usually placed in the task code word (at the beginning of the task control block).

roll screen. A display screen which is logically segmented into an optional history area and a work area. Output directed to the screen starts display at the beginning of the work area and continues on down in a line-by-line sequence. When the work area gets full, the operator presses ENTER/SEND and its contents are shifted into the optional history area and the work area itself is erased. Output now starts again at the beginning of the work area.

SBIOCB. See sensor based I/O control block.

second-level index block. In an indexed data set, the second-lowest level index block. It contains the addresses and high keys of several primary-level index blocks.

secondary file. See secondary index.

secondary index. For the Indexed Access Method Version 2, an indexed file used to access data records by their secondary keys. Sometimes called a secondary file.

secondary index entry. For the Indexed Access Method Version 2, this an an entry in the directory describing a secondary index.

secondary key. For the Indexed Access Method Version 2, the key used to uniquely identify a data record.

secondary option menu. In the Session Manager, the second in a series of predefined procedures grouped together in a hierarchical structure of menus. Secondary option menus provide a breakdown of the functions available under the session manager as specified on the primary option menu.

secondary task. Any task other than the primary task. A secondary task must be attached by a primary task or another secondary task.

secondary station. In a Series/1 to Series/1 attachment, the processor that is under the control of the primary station.

sector. The smallest addressable unit of storage on a disk or diskette. A sector on a 4962 or 4963 disk is equivalent to an Event Driven Executive record. On a 4964 or 4966 diskette, two sectors are equivalent to an Event Driven Executive record.

selection. In data communications, the process by which the multipoint control station asks a tributary station if it is ready to send messages.

self-defining term. A decimal, integer, or character that the computer treats as a decimal, integer, or character and not as an address or pointer to data in storage.

sensor based I/O control block (SBIOCB). A control block containing information related to sensor I/O operations.

sequential access. The processing of a data set in order of occurrence of the records in the data set. (1) In the Indexed Access Method, the processing of records in ascending collating sequence order of the keys. (2) When using READ/WRITE, the processing of records in ascending relative record number sequence.

serially reusable resource (SRR). A resource that can only be accessed by one task at a time. Serially reusable resources are usually managed via (1) a QCB and ENQ/DEQ statements or (2) an ECB and WAIT/POST statements.

service request. A device generated signal used to inform the GPIB controller that service is required by the issuing device.

session manager. A series of predefined procedures grouped together as a hierarchical structure of menus from which you select the utility functions, program preparation facilities, and language processors needed to prepare and execute application programs. The menus consist of a primary option menu that displays functional groupings and secondary option menus that display a breakdown of these functional groupings.

shared resource. A resource that can be used by more than one task at the same time.

shut down. See data set shut down.

source module/program. A collection of instructions and statements that constitute the input to a compiler or assembler. Statements may be created or modified using one of the text editing facilities.

spool job. The set of print records generated by a program (including any overlays) while engueued to a printer designated as a spool device.

spool session. An invocation and termination of the spool facility.

spooling. The reading of input data streams and the writing of output data streams on storage devices, concurrently with job execution, in a format convenient for later processing or output operations.

SRQ. See service request.

stand-alone dump. An image of processor storage written to a diskette.

stand-alone dump diskette. A diskette supplied by IBM or created by the $DASDI utility.

standard labels. Fixed length 80-character records on tape containing specific fields of information (a volume label identifying the tape volume, a header label preceding the data records, and a trailer label following the data records).

static screen. A display screen formatted with predetermined protected and unprotected areas. Areas defined as operator prompts or input field names are protected to prevent accidental overlay by input data. Areas defined as input areas are not protected and are usually filled in by an operator. The entire screen is treated as a page of information.

station. In BSCAM communications, a BSC line attached to the Series/1 and functioning in a point-to-point or multipoint connection. Also, any other terminal or processor with which the Series/1 communicates.

subroutine. A sequence of instructions that may be accessed from one or more points in a program.

supervisor. The component of the Event Driven Executive capable of controlling execution of both system and application programs.

system configuration. The process of defining devices and features attached to the Series/1.

SYSGEN. See system generation.

system generation. The processing of defining I/O devices and selecting software options to create a supervisor tailored to the needs of a specific Series/1 hardware configuration and application.

system partition. The partition that contains the root segment of the supervisor (partition number 1, address space 0).

talker. A controller or active device on a GPIB bus that is configured to be the source of information (the sender) on the bus.

tape device data block (TDB). A resident supervisor control block which describes a tape volume.

tapemark. A control character recorded on tape used to separate files.

task. The basic executable unit of work for the supervisor. Each task is assigned its own priority and processor time is allocated according to this priority. Tasks run independently of each other and compete for the system resources. The first task of a program is the primary task. All tasks attached by the primary task are secondary tasks.

task code word. The first two words (32 bits) of a task's TCB; used by the emulator to pass information from system to task regarding the outcome of various operations, such as event completion or arithmetic operations.

task control block (TCB). A control block that contains information for a task. The information consists of pointers, save areas, work areas, and indicators required by the supervisor for controlling execution of a task.

task supervisor. The portion of the Event Driven Executive that manages the dispatching and switching of tasks.

TCB. See task control block.

terminal. A physical device defined to the EDX system using the TERMINAL configuration statement. EDX terminals include directly attached IBM displays, printers and devices that communicate with the Series/1 in an asynchronous manner.

terminal control block (CCB). A control block that defines the device characteristics, provides temporary storage, and contains links to other system control blocks for a particular terminal.

terminal environment block (TEB). A control block that contains information on a terminal's attributes and the program manager operating under the Multiple Terminal Manager. It is used for processing requests between the terminal servers and the program manager.

terminal screen manager. The component of the Multiple Terminal Manager that controls the presentation of screens and communications between terminals and transaction programs.

terminal server. A group of programs that perform all the input/output and interrupt handling functions for terminal devices under control of the Multiple Terminal Manager.

terminal support. The support provided by EDX to manage and control terminals. See terminal.

timer. The timer features available with the Series/1 processors. Specifically, the 7840 Timer Feature card (4955 only) or the native timer (4952, 4954, and 4956). Only one or the other is supported by the Event Driven Executive.

trace range. A specified number of instruction addresses within which the flow of execution can be traced.

transaction oriented applications. Program execution driven by operator actions, such as responses to prompts from the system. Specifically, applications executed under control of the Multiple Terminal Manager.

transaction program. See transaction-oriented applications.

transaction selection menu. A Multiple Terminal Manager display screen (menu) offering the user a choice of functions, such as reading from a data file, displaying data on a terminal, or waiting for a response. Based upon the choice of option, the application program performs the requested processing operation.

tributary station. In BSCAM communications, the stations under the supervision of a control station in a multipoint connection. They respond to the control station's polling and selection.

unmapped storage. The processor storage in your processor that you did not define on the SYSTEM statement during system generation.

unprotected field. A field in which the operator can use the keyboard to enter, modify or erase data. Also called non-protected field.

update. (1) To alter the contents of storage or a data set. (2) To convert object modules, produced as the output of an assembly or compilation, or the output of the linkage editor, into a form that can be loaded into storage for program execution and to update the directory of the volume on which the loadable program is stored.

user exit. (1) Assembly language instructions included as part of an EDL program and invoked via the USER instruction. (2) A point in an IBM-supplied program where a user written routine can be given control.

variable. An area in storage, referred to by a label, that can contain any value during program execution.

vary offline. (1) To change the status of a device from online to offline. When a device is offline, no data set can be accessed on that device. (2) To place a disk or diskette in a state where it is unknown by the system.

vary online. To place a device in a state where it is available for use by the system.

vector. An ordered set or string of numbers.

volume. A disk, diskette, or tape subdivision defined using $INITDSK or $TAPEUT1.

volume descriptor entry (VDE). A resident supervisor control block that describes a volume on a disk or diskette.

volume label. A label that uniquely identifies a single unit of storage media.

Index

The following index contains entries for this book only. See the *Library Guide and Common Index* for a Common Index to all Event Driven Executive books.

Index

Index

Index

Index

Index

IBM Series/1 Event Driven Executive

Publications Order Form

Instructions:

1. Complete the order form, supplying all of the requested information. (Please print or type.)
2. If you are placing the order by phone, dial **1-800-IBM-2468.**
3. If you are mailing your order, fold the order form as indicated, seal with tape, and mail. We pay the postage.

Ship to:

Name: ______________________

____

Address: ______________________

____

City: ______________________

State: ______________ Zip: ______________

Bill to:

Customer number: ______________________

Name: ______________________

____

Address: ______________________

____

City: ______________________

State: ______________ Zip: ______________

Your Purchase Order No.: ______________________

Phone: (   ) ______________________

Signature: ______________________

Date: ______________________

Order:

Description	Order number	Qty.
Reference books:		
Set of the following six books. To order individual copies, use the following order numbers.	SBOF-1627	______
Communications Guide	SC34-0638	______
Extended Address Mode and Performance Analyzer User Guide	SC34-0591	______
Installation and System Generation Guide	SC34-0646	______
Language Reference	SC34-0643	______
Library Guide and Common Index	SC34-0645	______
Messages and Codes	SC34-0636	______
Operator Commands and Utilities Reference	SC34-0644	______
Guides and reference cards:		
Set of the following four books and reference cards. To order individual copies, use the following order numbers.	SBOF-1628	______
Customization Guide	SC34-0635	______
Event Driven Language Programming Guide	SC34-0637	______
Operation Guide	SC34-0642	______
Problem Determination Guide	SC34-0639	______
Language Reference Card	SX34-0165	______
Operator Commands and Utilities Reference Card	SX34-0164	______
Conversion Charts Reference Card	SX34-0163	______
Reference Card Envelope	SX34-0166	______
Set of three reference cards and storage envelope. (One set is included with order number SBOF-1627)	SBOF-1629	______
Binders:		
3-ring easel binder with 1 inch rings	SR30-0324	______
3-ring easel binder with 2 inch rings	SR30-0327	______
Standard 3-ring binder with 1 inch rings	SR30-0329	______
Standard 3-ring binder with 1 1/2 inch rings	SR30-0330	______
Standard 3-ring binder with 2 inch rings	SR30-0331	______
Diskette binder (Holds eight 8-inch diskettes.)	SB30-0479	______

Publications Order Form

Cut or Fold Along Line

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
1 Culver Road
Dayton, New Jersey 08810

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

International Business Machines Corporation

IBM Series/1 Event Driven Executive
Installation and System Generation Guide

Order No. SC34-0646-0

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

SC34-0646-0

Printed in U.S.A.

Reader's Comment Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Information Development, Department 28B
P.O. Box 1328
Boca Raton, Florida 33432

Fold and tape Please Do Not Staple Fold and tape

IBM®

IBM Series/1 Event Driven Executive
Installation and System Generation Guide

Order No. SC34-0646-0

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for systems analysts, programmers, and operators of IBM systems. You may use this form to communicate your comments about this publication, its organization, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you. Your comments will be sent to the author's department for whatever review and action, if any, are deemed appropriate.

Note: *Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.*

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

SC34-0646-0
Printed in U.S.A.

Reader's Comment Form

Cut or Fold Along Line

Fold and tape   Please Do Not Staple   Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS   PERMIT NO. 40   ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Information Development, Department 28B
P.O. Box 1328
Boca Raton, Florida 33432

Fold and tape   Please Do Not Staple   Fold and tape

IBM®

International Business Machines Corporation

SC34-0646-0

SC34-0646-0
Program Numbers: 5719-XS5, 5719-XX6
File Number: S1-34
Printed in U.S.A.