IBM

Series/1

SC34-0936-0

Event Driven Executive Installation and System Generation Guide

Version 6.0

Library Guide and Common Index SC34-0938	Installation and System Generation Guide SC34-0936	Operator Commands and Utilities Reference SC34-0940
Language Reference SC34-0937	Communications Guide SC34-0935	Messages and Codes SC34-0939
Operation Guide SC34-0944	Event Driven Language Programming Guide SC34-0943	APPC Programming Guide and Reference SC34-0960
Problem Determination Guide SC34-0941	Customization Guide SC34-0942	Internal Design LY34-0364

IBM

Series/1

SC34-0936-0

Event Driven Executive Installation and System Generation Guide

Version 6.0

Library Guide and Common Index SC34-0938	**Installation and System Generation Guide** **SC34-0936**	Operator Commands and Utilities Reference SC34-0940
Language Reference SC34-0937	Communications Guide SC34-0935	Messages and Codes SC34-0939
Operation Guide SC34-0944	Event Driven Language Programming Guide SC34-0943	APPC Programming Guide and Reference SC34-0960
Problem Determination Guide SC34-0941	Customization Guide SC34-0942	Internal Design LY34-0364

First Edition (October 1987)

Use this publication only for the purposes stated in the section entitled "About This Book."

Changes are made periodically to the information herein; any such changes will be reported in subsequent revisions or Technical Newsletters.

This material may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

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

This publication could contain technical inaccuracies or typographical errors. A form for readers' comments is provided at the back of this publication. If the form has been removed, address your comments to IBM Corporation, Information Development, Department 28B (5414), P. O. Box 1328, Boca Raton, Florida 33429-1328. IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation whatever. You may, of course, continue to use the information you supply.

Summary of Changes for Version 6.0

This document contains the following changes:

3151 Display Terminal

- Appendix C, "Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals" was updated to include information about configuring and connecting the 3151 display terminal.
- Throughout the book, updates were made to include the 3151 display terminal as applicable.

4234 Printer

- Appendix A, "System Definition Statements" was updated to include information about defining the 4234 printer to the system.
- Throughout the book, updates were made to include the 4234 printer as applicable.

4956 J and K Processors

- Chapter 2, "Determine if the Starter System Meets Your Needs" was updated to reflect support of the 4956 model J and K extended address mode processor.
- Chapter 3, "Install EDX on a 4952, 4954, 4955, or 4956 Processor" was updated to reflect support of the 4956 model J and K extended address mode processor.
- Chapter 5, "Generate a Tailored Operating System" was updated to reflect support of the 4956 model J and K extended address mode processor.
- Appendix A, "System Definition Statements" was updated to reflect support of the 4956 model J and K extended address mode processor.

System Partition Statements

- References to the SYSTEM statement have been replaced by the appropriate system partition statement: SYSPARTS, SYSPARMS, SYSCOMM, or SYSEND.

$INSTAL Installation Utility

- Chapter 3, "Install EDX on a 4952, 4954, 4955, or 4956 Processor" shows you how to use the $INSTAL utility to do certain steps of the installation procedure.
- Appendix G, "Alternative Installation Instructions" is a new appendix containing the alternative instructions for installing the system without using the $INSTAL utility. These steps were formerly in Chapter 3.

EDX Line Sharing Support

- Chapter 4, "Select Your Required Support" has been updated to reflect support of the line sharing function.
- Appendix A, "System Definition Statements" shows you how to use the new SHARING operand for the TERMINAL statement for 4201/4202/4224 and ACCA devices.
- Appendix C, "Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals" was updated to show how to attach line sharing devices.
- Appendix F, "Setting Up the 4201, 4202, and 4224 Printers" has been updated to show line sharing support.

XON/XOFF Support for ACCA Devices

- Appendix A, "System Definition Statements" shows you how to use the new PACING operand for the TERMINAL statement for ACCA devices.

Terminal Speed of 19.2 KB

- Throughout the book, updates for this function have been included where applicable.

S/1 Multiline Communication Controller

- Appendix A, "System Definition Statements" was updated to include two new parameters for the EXIODEV statement.

SDLC and Communication Common Services Support

- Chapter 3, "Install EDX on a 4952, 4954, 4955, or 4956 Processor" shows how to use SDLC and communication common services support.

Miscellaneous

- Throughout the book, updates were made to reflect migration to Version 6.0.
- Chapters 4 and 5, and Appendix E have been updated for changes to the $EDXDEF and $LNKCNTL data sets for applicable functions.
- Appendix D, "Supervisor Module Names" has been updated to include new or changed module names.
- Appendix E, "Work Sheets" was updated to reflect support of new functions and devices, and storage changes.
- Numerous editorial and usability changes were made throughout the book.

Contents

About This Book

This book describes how to install the Event Driven Executive (EDX) operating system on a Series/1 and how to tailor the operating system to meet your application and hardware needs.

Audience

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

How This Book Is Organized

This book contains 6 chapters and 6 appendixes:

- 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 EDX starter system, an IBM-supplied operating system.
- Chapter 3, "Install EDX on a 4952, 4954, 4955, or 4956 Processor" provides the step-by-step procedures for installing the Event Driven Executive system on your Series/1.
- Chapter 4, "Select Your Required Support" on page 4-1 describes how to select the support (hardware and software features) you need for your system.
- Chapter 5, "Generate a Tailored Operating System" provides the step-by-step procedures for generating a tailored operating system.
- Chapter 6, "Migrate to Version 6.0" provides the migration procedures you need to follow to convert from EDX Versions 1 and 2 to EDX Version 6.0.
- 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, "Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals" contains planning information to use in defining 3101, 3151, 3161, 3163, and 3164 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.
- Appendix F, "Setting Up the 4201, 4202, and 4224 Printers" contains information you need to set up the 4201,4202, and 4224 printers. Use this appendix in conjunction with your printer hardware manual.

- Appendix G, "Alternative Installation Instructions" contains alternative instructions for doing certain installation procedure steps without using the $INSTAL utility.

Aids in Using This Book

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

- A table of contents that lists the major headings in this book.
- In example screens where you must answer a system request, any responses you must make are shown in red.
- Worksheets that help you plan your system requirements.
- An index of the topics covered in this book.

Using the Enter and Attention Keys

This book uses the term "enter key" to mean the key that indicates that you have completed input to the screen and now you want the system to process that data. This book uses the term "attention key" to mean the key that indicates that you want to direct keyboard input to the operating system supervisor. If your keyboard does not have these keys, use the corresponding keys on your keyboard.

A Guide to the Library

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

Contacting IBM about Problems

You can inform IBM of any inaccuracies or problems you find with 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, refer to the *IBM Series/1 Software Service Guide*, GC34-0099.

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.

You receive EDX 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 basic system matches your application needs, you can install it as your operating system. If the basic system does not match your needs, you must generate an operating system tailored to your needs.

What Do You Need To Do?

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

- Migrate to Version 6.0 if you have Version 1 or 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).

Migrate to Version 6.0 (Version 1 and 2 Only)

Before installing EDX Version 6.0, you must convert Version 1 and 2 data sets to make them compatible with Version 6.0 data sets. Chapter 6, "Migrate to Version 6.0" describes these conversion procedures.

Note: If you are a current user of EDX Version 3 or above and are installing EDX Version 6.0, you do not have to perform a special conversion procedure. Version 3 and above programs and data sets are compatible with Version 6.0. However, we suggest that you create a backup copy of your current system and pertinent data before installing EDX Version 6.0.

Determine if the Starter System Meets your Requirements

The information in Chapter 2 helps you decide if the IBM-supplied starter 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.

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

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 6.0 (Version 1 and 2 users only).
- Perform an initial program load (IPL) of 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.

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.

You can use the following work sheets to help you keep track of your selections:

Work Sheet 1 to estimate the size of your supervisor;

Work Sheet 2 to define I/O devices to the supervisor;

Work Sheet 3 to select your software support;

Work Sheet 4 to estimate 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 that is part of the starter system. This process consists of the following tasks:

1. Creating a set of system definition statements to reflect the hardware configuration of your system.
2. Selecting the supervisor object modules required to support your I/O devices and system features.
3. Assembling the system definition statements created in task 1 above.
4. Link editing the object modules produced in task 3 with the supervisor object modules selected in task 2 to produce a modified supervisor.
5. Initializing your new supervisor.
6. Performing an IPL of your system.
7. Verifying your operating system.

Chapter 2. Determine If the Starter System Meets Your Needs

The information in this chapter helps you determine if the configuration of the starter 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 features that are built into the starter system.

If you have any devices that are not built into the starter 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 are not built into the starter system (such as floating point, spooling, or timers).

Tailoring your operating system means building your needs into the operating system. You can add support for certain devices and software features that are not included in the starter system, and you can delete support for devices or 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.

What Is the Starter System?

The starter 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 starter 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. However, you should not use these modules in a production environment. If your hardware configuration is not supported by the starter 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 IDSK 5.25-inch disk

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 IDSK diskette unit at address X'61'

One 4964 diskette storage unit (at address X'02')

Two 4965 diskette storage units (one at address X'44' and one at address X'45')

One 4966 diskette storage unit (at address X'22')

One 4978 (at address X'24'), 4979 (at address X'04'), or 4980 terminal (at address X'80')

One Teletypewriter [1] (TTY) device (at address X'00')

One 4974 printer (at address X'01')

One Multifunction Attachment (MFA) - Feature Number #1310 (at address X'58')

One Multidrop Work Station Attachment - Feature Number #1250 (at address X'80')

One 4975-01L printer (at address X'5A') and one 4975-02L printer (at address X'5B') attached through the Multifunction Attachment

One 3101 terminal (block mode) or 3151/3161/3163/3164 in 3101 emulation mode attached at 9600 bits per second through the Multifunction Attachment (at address X'59').

Notes:

1. If there is a Multifunction Attachment at address X'58', the starter system expects a 3101 terminal or its equivalent in block mode locally attached at 9600 bits per second through the Multifunction Attachment at address X'59'. (For information on customizing adapters, see Appendix B, "Customizing Adapters with Hardware Jumpers.")
2. If there is no Multifunction Attachment at address X'58', $EDXNUC expects a 3101 terminal or its equivalent in block mode attached point-to-point at 9600 bits per second and range high through adapter #1610 (at address X'08').
3. If there is no #1610 adapter at address X'08', $EDXNUC expects a 3101 terminal or its equivalent in block mode attached through adapter #2091/2092 (at address X'60').
4. If there is an IDSK disk at address X'60', the 2091/2092 is not recognized.
5. If there is no #2091/2092 adapter at address X'60', $EDXNUC expects a 3101 terminal or its equivalent in block mode attached through adapter #2091/2092 (at address X'90').
6. If there is no #2091/2092 adapter at address X'60' or X'90', $EDXNUC expects a 3101 terminal or its equivalent in block mode attached through adapter #2095/2096 (at X'68').

---

[1] Trademark of Teletype Corporation

Software Features Supported by $EDXNUC

$EDXNUC provides the following software support:

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

Devices Not Supported by $EDXNUC

The Series/1 devices in the checklist on the following page are ***not*** supported by the basic starter system ($EDXNUC). Use this checklist 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		
4201, 4202, or 4224 printer		
4973 printer		
5219 printer		
5224 printer		
4234, 5225, or 5262 printer		
2741 communications terminal		
3151/3161/3163/3164 terminals (in their native mode)		
General Purpose Interface Bus		
Binary Synchronous Line		
Local Communications Controller attachment		
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 through feature #1610, #2091/2092, or #2095/2096		
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 2-1. Checklist of Devices Not Supported by the Starter System ($EDXNUC)

You also need to generate a tailored operating system if your system contains more than one of the following devices:

Device	Yes	No
IDSK disk unit		
IDSK diskette unit		
4962 disk		
4963 disk		
4967 disk		
30-megabyte disk		
60-megabyte disk		
4964 or 4966 diskette units		
4978, 4979, or 4980 terminal		
3101 or 3151/3161/3163/3164 terminal in 3101 emulation mode		
TTY		

Figure 2-2. 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		
Local Communications Controller attachment		
Dynamic data set extent support		
EXIO support		
EXIO trace option		
Unmapped storage support		
Host communications support		
ACCA trace option		

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

Software Features	Yes	No
Indexed Access Method QCBs		
Translation tables for 2741 terminal		
Timer support for the 4955 processor		
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 2-3		
1024 bytes/sector IPL capability and/or I/O support		
Interprogram communications through virtual terminals		
Spooling support		
EBCDIC/floating-point conversion		
Floating-point arithmetic		
Remote Management Utility		
Interactive debug		
Virtual Terminal option		

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

Chapter 3. Install EDX on a 4952, 4954, 4955, or 4956 Processor

This chapter describes how to install the Event Driven Executive (EDX) on a Series/1 with a 4952, 4954, 4955, or 4956 processor.

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-XS6).
  - Any of the following program products depending on your installation requirements:
    - Program Preparation Facility (5719-XX7)
    - EDX Macro Library (5719-LMA)
    - EDX Macro Library/Host (5740-LM7).
- A Series/1 with one of the following minimum configurations (each configuration corresponds to the type of processor installed on your system):

  One Processor (one of the following):

  - 4952, 4954, 4955 (except models A and C), or 4956 processor with 96KB 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):

  - IDSK disk.
  - 4962 disk storage unit.
  - 4963 disk subsystem.
  - 4967 disk subsystem.
  - 30-megabyte disk storage unit (DDSK-30).
  - 60-megabyte disk storage unit (DDSK-60).

  One IDSK diskette, 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 terminal
  - 4980 terminal attached through the Multidrop Work Station Attachment Feature #1250 at 100K bits per second.
  - 3101 terminal (character mode) or equivalent device attached through the Teletypewriter Adapter Feature #7850 at 9600 bits per second.

- 3101 terminal (block mode) or equivalent device attached point-to-point through Feature #1610, #2091/2092, or #2095/2096 at 9600 bits per second and range high.
- 3101 terminal (block mode) or equivalent device attached through a Multifunction Attachment Feature #1310 at 9600 bits per second in RS-422-A mode.

In addition, the following addresses ***must*** contain the listed devices (or none at all) for the starter system:

Device	Type	Address (Hex)
IDSK	diskette unit	61
4964	diskette unit	02[1]
4965	diskette drive	44[2]
4966	diskette magazine unit	22
4974	printer	01
4975-01L	printer (through MFA)	5A
4975-02L	printer (through MFA)	5B
4978	display station	24
4979	display station	04
4980	display station	80[3]
#1250	Multidrop Work Station Attachment	80[3]
#1610	Asynchronous Communications Single-Line Adapter	08[4]
#2091/2092	Asynchronous Communications Multiline Adapter	60 or 90[4,7]
#2095/2096	Feature-Programmable Communications Adapter	68[4]
#7850	Teletypewriter Attachment	00[5]
#1310	Multifunction Attachment (MFA)	58[6]

Figure 3-1. Address Assignments for Minimum Series/1 Configuration

Notes:

1. Do not assign a 4965 or 4966 to hardware address X'02' if you have a 4964 at that address (as with the starter system). If you do, any attempt to IPL a diskette from the 4965 or 4966 causes the system to destroy any data on that diskette.
2. If the diskette unit is part of a 30D/60D/60E processor, it must be at address X'45'.
3. Supports 4980 at ADDRESS=80, PORT=0, and SECADDR=AA on the Multidrop Work Station Adapter #1250.
4. Supports 3101 models 20, 22, 23 and 3151/3161/3163/3164 (in 3101 emulation mode) in block mode at 9600 bits per second.
5. Supports 3101 models 10, 12, 13, 20, 22, 23 and 3151/3161/3163/3164 (in 3101 emulation mode) in character mode at 9600 bits per second.

6. Supports 3101 model 23 and 3151/3161/3163/3164 (in 3101 emulation mode) in block mode at address X'59' at 9600 bits per second.
7. If you are using an IDSK diskette at address X'61', use #2091/2092 at address X'90'.

Preparing to Install EDX

Note: This discussion is limited to the Basic Supervisor and Emulator (5719-XS6) and the Program Preparation Facility (5719-XX7). The Macro Library (5719-LMA) and Macro Library/Host (5740-LM7) licensed programs are not addressed.

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

- IDSK disk
- 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 2CA/2CB
- 4967 model 3CA/3CB
- DDSK-30
- DDSK-60.

An EDX supervisor must reside in a data set named $EDXNUCx (where x is any alphanumeric character you wish to specify). As shipped from IBM, diskette XS6001 contains a supervisor called the starter system in a data set named $EDXNUC. The first step in installation requires that you perform an initial program load (IPL) of the starter system from diskette XS6001. Therefore, the Series/1 must have one of the following devices wired as either the primary or alternate IPL device:

- IDSK diskette unit at hardware address X'61"
- 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 purpose 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.

- IDSK disk at hardware address X'60'
- 4962 disk (any model)
- 4963 disk (any model)

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

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 terminal at hardware address X'24'
- 4979 terminal at hardware address X'04'
- 4980 terminal at hardware address X'80'
- 3101B terminal or equivalent device at hardware address X'59'.

Also, the Series/1 may have a 4974 printer at hardware address X'01', a 4975-01L printer at hardware address X'5A', or a 4975-02L printer at hardware address X'5B'. If you have a 4973 printer, after installation is complete 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 printer.

Installing EDX

The procedures and instructions for installing EDX 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 6.0."

Current users of EDX Version 3 or later do not need to perform a special migration procedure. EDX programs for Version 3 or later are compatible with EDX Version 6.0. However, we suggest that you create a backup copy of your current system before installing EDX Version 6.0.

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 $EDXDEF, LINKCNTL, or SUPPREPS?

Step 2 IPL the starter system from diskette XS6001.

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 or later, 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 3-8 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 using the $INSTAL utility.

Step 5 IPL the starter system from disk.

Step 6 Copy the system support modules and production utilities using the $INSTAL utility.

Step 7 Copy the program preparation modules and utilities using the $INSTAL utility (this step is required if the Program Preparation Facility 5719-XX7 is being installed).

Step 8 Exercise utilities and program preparation facilities.

Step 9 Use the shared SDLC and Communication Common Services support.

Step 10 Use the APPC support.

EDX Version 6.0 is shipped on both 8-inch and 5.25-inch starter diskettes. If you want to list the contents of these starter diskettes, use the $DISKUT1 utility as described in the *Operator Commands and Utilities Reference*.

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

5719-XS6 Basic Supervisor and Emulator

Diskette XS6001 contains the IBM-supplied supervisor program ($EDXNUC) and the utilities necessary to install the product.

Diskette XS6002 contains utilities that are included with the supervisor.

Diskette XS6003 contains utilities that are included with the supervisor.

Diskette XS6004 contains the remaining utilities included with the supervisor.

Diskette XS6005 contains modules that are link edited with application programs. These are object modules that support various system functions.

Diskette XS6006 contains supervisor object modules that are accessed only during the system generation process.

Diskette XS6007 contains modules that are required for Communication Common Services and shared SDLC.

Diskette XS6008 contains modules required for APPC support.

Diskette XS6009 contains the stand-alone dump program used for APAR reporting.

5719-XX7 Program Preparation Facility

Diskette XX7001 contains the EDX program preparation modules.

Diskette XX7002 contains copy modules for inclusion in user application programs.

Diskette XX7003 contains copy modules for inclusion in user application programs.

Diskette XX7004 contains the program preparation utilities and the session manager.

You are ready now to install the Event Driven Executive.

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
```

  In this example, *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 6.0" 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.

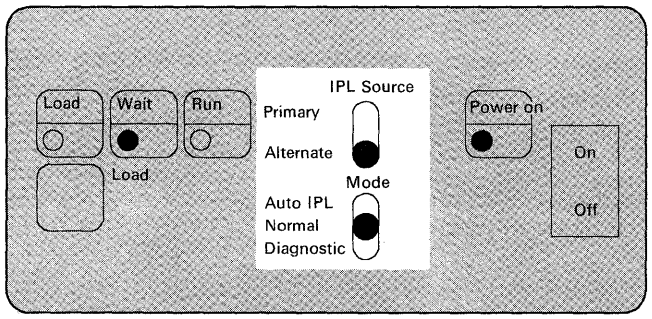
Step 2 - IPL the Starter System

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.

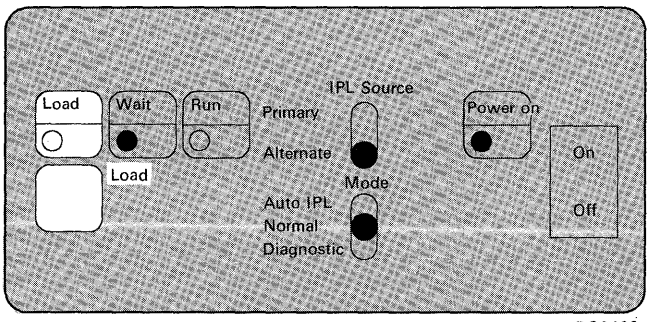


BG0406

2(b). Insert diskette XS6001 into the diskette unit.

EDX supports the 4964, 4965, and IDSK diskette units and the 4966 diskette subsystem. If you have the 4966 or IDSK unit, you need to vary the diskette online. If you have the 4964 or 4965 unit, the vary on will be done automatically when you close the diskette drive door. 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.



BG0409

When the IPL is complete, the system displays an IPL message on the logging device. The actual message varies according to the physical storage available on your system.

```
*** EVENT DRIVEN EXECUTIVE *** V6.0
***          XPS          ***
IPL = $EDXNUC,XS6001

              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   27904  37632     0      0     6D00   6D00   9300  10000
  2   24576  40960     0      0     6000   6000   A000  10000
  3       0  65536     0      0        0      0   A000  10000
  4       0  65536     0      0        0      0  10000  10000
  5       0  65536     0      0        0      0  10000  10000
  6       0  65536     0      0        0      0  10000  10000
  7       0  65536     0      0        0      0  10000  10000
  8       0  65536     0      0        0      0  10000  10000
TOTAL SIZES (HEX):                  CD00         73300
UNMAPPED STORAGE =    1793 (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).

If you are a current user of EDX Version 3 or later, you do not need to initialize these volumes; they already exist on your system. However, you must ensure that the volumes are large enough to install EDX Version 6.0. 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 minimum size (in records) for each volume is as follows:

Volume	Size
EDX002	15,000
ASMLIB	12,000
EDX003	16,000
MACLIB	44,000

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

3(a). Load the utility $INITDSK:

1. Press the attention key or its equivalent.
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         nnP, LP = nnnn, PART=x
$INITDSK - DISK INITIALIZATION UTILITY

COMMAND (?):
```

3(b). Initialize the disk.

(If you do not wish to initialize your disk, skip to step 3(c)).

The starter system looks for a disk at address X'48'. If your system has no disk at that address, you must power off all other disks that have an address lower than the one you are using.

For example, 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. If you are using an IDSK disk at address X'60' and you have a 4962 at address X'03' or a 4967 at address X'58', you must power off the devices with address lower than X'60'.

1. Enter **ID nn**, where *nn* is the hexadecimal address of the disk you are initializing. Use the $IOTEST utility to determine the device address.
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 fixed-head disk)
VOLUME: EDX002
SIZE IN RECORDS: 15000
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 fixed-head disk)
VOLUME:  ASMLIB
SIZE IN RECORDS:  12000
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 fixed-head disk)
VOLUME:  EDX003
SIZE IN RECORDS:  16000
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
```

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 fixed-head disk)
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 Using $INSTAL

Use the $INSTAL utility to copy the basic EDX 6.0 starter system, along with the basic utilities and support modules, from XS6001 to the disk volume EDX002.

4(a). Load the utility $INSTAL:

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

```
> $L $INSTAL
LOADING $INSTAL     nnP, LP=nnnn, PART=x

   **** SYSTEM MAINTENANCE UTILITY  ****

ENTER THE CURRENT DATE (MM/DD/YY): mm/dd/yy

COMMAND (?): IN

ENTER THE HISTORY FILE (DSNAME,VOL) OR 'END':  $HISTXS6,EDX002

$HISTXS6 ON EDX002 DID NOT EXIST BUT HAS BEEN ALLOCATED

ENTER THE $INSTAL CONTROL DATA SET (DSNAME,VOL):  $$INXS6A

USE THE TARGET VOLUMES SPECIFIED
IN THE $INSTAL CONTROL DATA SET (Y/N)? Y

COPYING XS6001 TO EDX002...

(Each module name is listed as it is copied)

XS6001 COPY COMPLETE

...INSTALLATION COMPLETED...

COMMAND (?): EN

$INSTAL ENDED
```

Note: Respond to the prompts as stated above unless you do not want to use the target volume EDX002. To use a different target volume, enter N in response to the USE THE TARGET VOLUMES prompt, and you will be prompted to enter the desired volume name.

If you are a user of Version 3 or a later version and did not reinitialize the volumes required to install EDX Version 6.0 (specifically, EDX002, ASMLIB, EDX003, and FHVOL, if applicable), you need to initialize the starter system using the II command of the $INITDSK utility. Complete the following step before proceeding to Step 4(c).

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

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

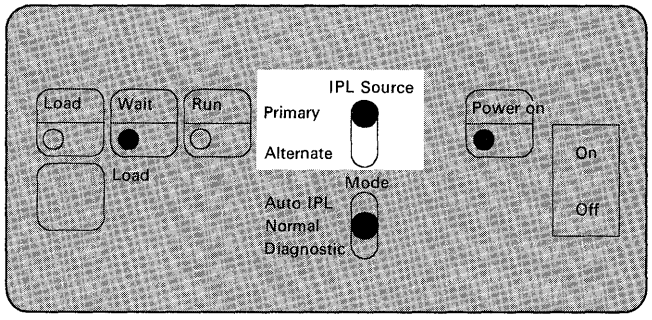
```
> $L $INITDSK
LOADING $INITDSK   nnP, LP= nnnn, PART=x

$INITDSK - DISK INITIALIZATION UTILITY

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

COMMAND (?):  EN
$INITDSK ENDED
```

4(c). Remove diskette XS6001 from the diskette unit.

Step 5 - IPL the Starter Supervisor from Disk

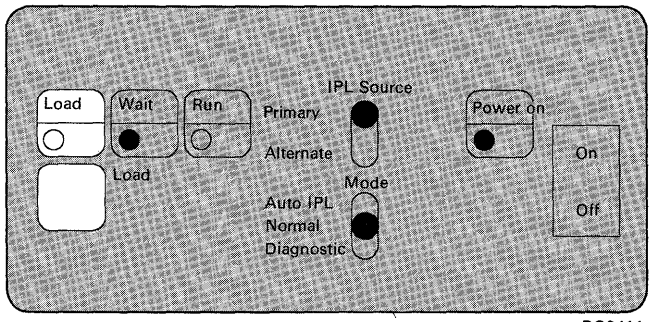
5(a). Set the IPL Source switch to IPL from the disk unit.

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



BG0410

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



BG0411

When the IPL is complete, the system issues a message similar to the one below indicating that the starter system is installed successfully.

```
*** EVENT DRIVEN EXECUTIVE ***  V6.0
***         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   27904  37632      0      0   6D00   6D00    9300  10000
  2   24576  40960      0      0   6000   6000    A000  10000
  3       0  65536      0      0      0      0   10000  10000
  4       0  65536      0      0      0      0   10000  10000
  5       0  65536      0      0      0      0   10000  10000
  6       0  65536      0      0      0      0   10000  10000
  7       0  65536      0      0      0      0   10000  10000
  8       0  65536      0      0      0      0   10000  10000
TOTAL (HEX):                       CD00        73300
UNMAPPED STORAGE =  1793 (DEC) 2K BLOCKS

$LOG AUTOLOAD STARTED
$LOG AUTOLOAD COMPLETE
EDX INITIALIZATION COMPLETE

LOGGING ACTIVATED
```

Step 6 - Copy the Production Utilities and System Support Modules Using the $INSTAL Utility

Use the $INSTAL utility to copy the production utilities from XS6002, XS6003, and XS6004, and the system support modules from XS6005.

The $INSTAL control data set is $$INXS6B on EDX002. This data set tells $INSTAL which volumes to copy and where to copy them. Unless otherwise specified, $INSTAL will use default volumes and copy XS6002, XS6003, and XS6004 to EDX002, and XS6005 to ASMLIB. If you are using volume names other than EDX002 (the IPL volume) and ASMLIB, you need to edit the $INSTAL control data set to reflect the names you are using. For more information on using the $INSTAL edit function, refer to the *Operator Commands and Utilities Reference*.

6(a). Insert diskette XS6002 into the diskette unit. The contents of this diskette will be copied to EDX002. If you are using a 4966 Diskette Magazine unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

To vary on the diskette do the following:

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

For the 4966:

```
> $VARYON 22,1
XS6002 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XS6002 ONLINE
```

6(b). Press the attention key and load the $INSTAL utility with the $L operator command. Specify the IN function as shown below.

```
> $L $INSTAL
LOADING $INSTAL          nnP, LP=nnnn, PART=x

  ****  SYSTEM MAINTENANCE UTILITY ****

 ENTER THE CURRENT DATE (MM/DD/YY): mm/dd/yy

COMMAND (?): IN
```

6(c). When prompted, enter the name and volume of the history file for the utilities. Use the same name and volume you specified in Step 4(a). Respond to the prompts as shown.

```
ENTER THE HISTORY FILE (DSNAME,VOL) OR 'END': $HISTXS6

ENTER THE $INSTAL CONTROL DATA SET (DSNAME,VOL): $$INXS6B

USE THE TARGET VOLUMES SPECIFIED
IN THE $INSTAL CONTROL DATA SET (Y/N)? Y

 COPYING XS6002 TO EDX002...

 (Each module name is listed as it is copied)

 XS6002 COPY COMPLETE
```

6(d). Replace diskette XS6002 with diskette XS6003. If you are using a 4966 or IDSK diskette drive you must vary it online as shown in Step 6(a). If you do not have one of these types of drives, the diskette will be automatically varied on when you close the drive door.

Do the following:

1. Press the attention key.
2. Enter GO to proceed.

```
INSERT THE DISKETTE LABELED XS6003 IN THE DISKETTE DRIVE.
CLOSE THE DISKETTE DRIVE AND DO A '$VARYON' FOR THAT ADDRESS.
WHEN $VARYON IS COMPLETED, PRESS THE <ATTN> KEY AND ENTER 'GO'.

> $VARYON 61
XS6003 ONLINE
> GO

 COPYING XS6003 TO EDX002...

 (Each module name is listed as it is copied)

 XS6003 COPY COMPLETE
```

6(e). $INSTAL will continue with this procedure until all the required diskettes are copied. After the last diskette has been copied, the following message will appear:

```
 ...INSTALLATION COMPLETED...

COMMAND (?):
```

6(f). Remove the last diskette from the diskette unit.

- If you are going to create a tailored operating system, develop application programs on your Series/1, or use the session manager, you need to copy the Program Preparation modules. Proceed to Step 7.
- If not, end $INSTAL by responding to the COMMAND (?): prompt as shown and proceed to Step 8.

```
COMMAND (?):  EN

 $INSTAL ENDED
```

Step 7 - Copy the Program Preparation and Session Manager Modules (5719-XX7) Using the $INSTAL Utility

This step copies the program preparation modules and utilities and is required only if you are installing 5719-XX7 (the Program Preparation Facility). You need these modules to develop application programs or to use the session manager on your Series/1.

Note: Either the Program Preparation Facilities (5719-XX7) 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.

Use the $INSTAL utility to copy the program preparation modules from XX7001, XX7002, and XX7003 to ASMLIB, and the program preparation utilities and the session manager from XX7004 to EDX002.

If you are using volume names other than listed here, you need to edit the $INSTAL control data set to reflect the names you are using. For more information on using the $INSTAL edit function, refer to the *Operator Commands and Utilities Reference*.

7(a). Insert diskette XX7001 into the diskette unit. The contents of this diskette will be copied to ASMLIB. If you are using a 4966 Diskette Magazine unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

To vary on the diskette do the following:

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

For the 4966:

```
> $VARYON 22,1
XX7001 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XX7001 ONLINE
```

7(b). If you ended the $INSTAL utility in Step 6(f), you must load $INSTAL again. See Step 6(b) for information on loading $INSTAL. Issue the IN install command and respond to the prompt as shown:

```
COMMAND (?): IN
```

7(c). Enter your history file name and volume for the 5719-XX7 product when prompted. If the history file does not exist, it will be allocated for you. You can select any valid file name for this file. The $INSTAL control data set is $$INXX7 on XX7001.

```
ENTER THE HISTORY FILE (DSNAME,VOL) OR 'END': $HISTXX7

ENTER THE $INSTAL CONTROL DATA SET (DSNAME,VOL): $$INXX7,XX7001

USE THE TARGET VOLUMES SPECIFIED
IN THE $INSTAL CONTROL DATA SET (Y/N)? Y

 COPYING XX7001 TO ASMLIB...

 (Each module name is listed as it is copied)

 XX7001 COPY COMPLETE
```

7(d). Replace diskette XX7001 with diskette XX7002. If you are using a 4966 or IDSK diskette drive you must vary it online as shown in Step 6(a).

Do the following:

1. Press the attention key.
2. Enter GO to proceed.

```
INSERT THE DISKETTE LABELED XX7002 IN THE DISKETTE DRIVE.
CLOSE THE DISKETTE DRIVE AND DO A '$VARYON' FOR THAT ADDRESS.
WHEN $VARYON IS COMPLETED, PRESS THE <ATTN> KEY AND ENTER 'GO'.

> $VARYON 61
XX7002 ONLINE
> GO

 COPYING XX7002 TO ASMLIB...

 (Each module name is listed as it is copied)

 XX7002 COPY COMPLETE
```

7(e). $INSTAL will continue to prompt for diskettes as it completes copying them until all the 5719-XX7 diskettes are copied. When the installation is completed, the following message will appear:

```
 ...INSTALLATION COMPLETED...
```

7(f). Remove the last diskette from the diskette unit.

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

```
 COMMAND (?): EN

 $INSTAL  ENDED
```

- 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-LMA program directory.

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. 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. 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.

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

```
> $L $DISKUT1
LOADING $DISKUT1      nnP, LP= nnnn, PART=x
$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
```

- 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 XS6001 to the wrong volume. **Repeat Steps 3 through 7.**

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

```
> $L $EDXASM,ASMLIB CALCSRC ASMWORK ASMOBJ
LOADING $EDXASM      nnP, LP= nnnn, PART=x

SELECT OPTIONS (?):  NOLIST END
ASSEMBLY STARTED    3 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
```

- 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 XX7001 through XX7004 to the wrong volumes. **Repeat Steps 3 through 7.**

8(c). Press the attention key and load **$EDXLINK** using the data set shown in the example. 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     nnP, LP= nnnn, PART=x
$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
```

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
LOADING CALCDEMO     nP, LP=nnnn, PART=x
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
```

Step 9 - Using the Shared SDLC and Communication Common Services Support

The following procedure is optional and should be used only if you want shared SDLC and Communication Common Services support. This support is needed for systems using Advanced Program-to-Program Communications (APPC) and Primary Systems Network Architecture (PSNA) support, or for SNA using shared SDLC.

You will need to install the shared SDLC and Communication Common Services support on diskette XS6007 using the $INSTAL utility.

Refer to *IBM Series/1 Event Driven Executive Systems Network Architecture and Remote Job Entry Guide*, SC34-0773 and *IBM Series/1 Event Driven Executive Primary Systems Network Architecture Programming Guide*, SC34-0762 for more information about shared SDLC and Communication Common Services support.

9(a). Use the $INSTAL utility to install shared SDLC and Communication Common Services.

Press the attention key and load $INSTAL. Respond to the $INSTAL prompts as shown. (See Step 6 for information on using $INSTAL).

```
> $L $INSTAL
LOADING $INSTAL     nnP, LP=nnnn, PART=x

   **** SYSTEM MAINTENANCE UTILITY  ****

ENTER THE CURRENT DATE (MM/DD/YY): mm/dd/yy

COMMAND (?): IN

ENTER THE HISTORY FILE (DSNAME,VOL) OR 'END':  $HISTXS6

ENTER THE $INSTAL CONTROL DATA SET (DSNAME,VOL):  $$INXS6D,XS6007

USE THE TARGET VOLUMES SPECIFIED
IN THE $INSTAL CONTROL DATA SET (Y/N)? Y

COPYING XS6007 to EDX002...
```

Note: This is the same history file you have been using for the rest of 5719-XS6. SDLC will show up as an option.

Step 10 - Using the Advanced Program-to-Program Communications Support

The IBM-supplied diskette XS6008 contains the Advanced Program-to-Program Communication (APPC) support. Installation of this diskette is optional. Refer to the *Advanced Program-to-Program Communication Programming Guide and Reference* for installation procedures.

Now that you have installed EDX, you can generate a tailored operating system or prepare application programs for execution on your Series/1. To generate a tailored operating system, see Chapter 4, "Select Your Required Support." For information on preparing programs using the Event Driven Language (EDL), 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 terminal at address X'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 terminal at hardware address X'04'. If there is no terminal at hardware address X'04', terminal initialization support relocates $SYSLOG to a 4978 terminal at hardware address X'24'. If there is no terminal at hardware address X'24', terminal initialization support relocates $SYSLOG to a 4980 terminal 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 terminal in block mode at hardware address X'59', connected through 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 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 terminal in block mode, connected to a Multiline Terminal Adapter (2091/2092) at hardware address X'60'.

Note: If you are using an IDSK diskette at device address X'60', the 2091/2092 will not be recognized.

If no MLTA attachment exists at hardware address X'60', terminal initialization support relocates $SYSLOGB to a 3101 terminal in block mode connected to a Feature Programmable Communications Adapter (2095/2096) at hardware address X'68'.

Chapter 4. Select Your Required Support

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.

Designing a Tailored Operating System

To design a tailored operating system, perform the following steps:

Step 1 Estimate your supervisor's storage requirements.

Step 2 Define your system configuration.

Step 3 Select your software support.

Step 4 Estimate the size for the dynamic data set extents table (DMEXTBL).

Step 5 Define the structure of your supervisor, if applicable.

Note: Be sure to review the program directories for all EDX-related products you are installing in case there are any special considerations for generating an EDX tailored operating system.

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

- 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."

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 65536 bytes) that is used for execution of programs. You can define up to 32 partitions for the 4956 models J and K, up to sixteen partitions for the 4956 models E, 60E, or H (with extended address mode defined), up to eight partitions for the 4955 or 4956 models B, 30D, 60D, and G, up to four partitions for the 4954, and up to two partitions for the 4952.

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 1024K bytes of storage. The devices supported and their address assignments are:

Device	Address (hexadecimal)
4962-1F Disk Storage Unit	03
4964 Diskette Unit	02
4978 terminal	24
4979 terminal	04
4973 printer	21

The 4979 terminal is defined as the primary system logging device ($SYSLOG) to receive all exception messages. The 4978 terminal is defined as the alternate system logging device ($SYSLOGA). The 4973 printer is defined as the system printer ($SYSPRTR) to receive the hard-copy output from all system programs.

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

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
Number of partitions	5
Size of partitions	(5,24,16,20,32)
Common area	Partitions 2, 4, and 5
Number of programs per partition	(1,10,10,5,5)
Unmapped storage	YES

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 5 - Defining Supervisor Structure" on page 4-33. 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     nnP, LP=nnnn, PART=x

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 DISKETTE UNIT
  48         3106       4963 DISK SUBSYSTEM
COMMAND (?):
```

Figure 4-1 on page 4-5 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 4-1 on page 4-5 shows the devices supported by the starter system $EDXNUC.

By comparing the previous figure and Figure 4-1 on page 4-5, 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
  59         2816       MFA (1310) SINGLE LINE ACCA    MODE = 3101B
  5A         0208       MFA WITH 4975-01L PRINTER
  5B         020A       MFA WITH 4975-02L PRINTER
  61         2202       IDSK DISK(ETTE)

COMMAND (?):
```

Figure 4-1. 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 communications line
- DISK - Defines direct access storage devices
- EXIODEV - Defines EXIO interface devices
- HOSTCOMM - Defines host communication support
- LCC - Defines a Local Communications Controller attachment
- SENSORIO - Defines sensor I/O devices
- SYSMSG - Defines where system messages are directed
- SYSPARTS - Defines the partitions in the system
- SYSPARMS - Defines the sizes of system buffers
- SYSCOMM - Defines partitions to be used as "base" and as "common"
- SYSEND - Defines the end of the system partition statements
- TAPE - Defines tape devices
- TERMINAL - Defines terminals
- TIMER - Defines system timer feature.

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.

SYSPARTS Statement

The SYSPARTS system partition statement defines the number of partitions, the number of programs allowed per partition, and the size of each partition for the system. The SYSPARTS statement must be specified only once and must be the first statement in $EDXDEF.

The NUMPART= operand specifies the number of partitions in the system. The range for NUMPART is 1 to the maximum number of partitions supported by your processor.

The PARTS= operand specifies the number of 2K (1K=1024 bytes) blocks of storage assigned to each partition. The range for PARTS is 0 to 32. 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.

The MAXPROG= operand specifies the maximum number of programs that can run concurrently in each partition.

See "SYSPARTS - Define Partitions" on page A-22 for a detailed description of this statement and the other operands.

If you determined in "Step 1 - Estimating Total Supervisor Size" on page 4-3 that your supervisor is 64K or greater or that you want to free up storage in partition 1, you may need to modify the NUMPART=, PARTS= and MAXPROG= operands of the SYSPARTS statement. After you perform "Step 5 - Defining Supervisor Structure" on page 4-33, 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. The number of blocks of storage assigned to each partition is 5, 24, 16, 20, and 32, respectively. The maximum number of programs that can run concurrently in each partition is 1, 10, 10, 5, and 5, respectively. The SYSPARTS statement defined for the sample system is:

```
SYSPARTS  NUMPART=5,PARTS=(5,24,16,20,32),                C
      MAXPROG=(1,10,10,5,5)
SYSCOMM   COMMON=(0,2,0,2,1)
SYSEND
```

The sample system logically and physically maps in storage as follows (an explanation follows the mapping examples):

Logical mapping:

Address space 0	22KB supervisor	24KB user space (partition 1)	Invalid
Address space 1	4KB COMMON	48KB user space (partition 2)	Invalid
Address space 2	32KB user space (partition 3)	Invalid	
Address space 3	4KB COMMON	40KB user space (partition 4)	Invalid
Address space 4	2KB COMMON	62KB user space (partition 5)	
Address space 5	Invalid (partition 6)		
Address space 6	Invalid (partition 7)		
Address space 7	11KB supervisor	1KB user space (partition 8)	Invalid
	784KB unmapped storage		

A0936001

Physical mapping:

46KB (part #1)	48KB (part #2)	32KB (part #3)
40KB (part #4)	62KB (part #5)	
12KB (part #8)	784KB unmapped	

A0936009

1. Partition 1 has 5 blocks (10KB) of actual storage mapped for user space. 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 has 24 blocks (48KB) of actual storage mapped for user space. 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 has 20 blocks (40KB) of actual storage mapped for user space. Partition 4 can execute up to 5 programs concurrently.
5. Partition 5 has 32 blocks (64KB) of actual storage mapped for user space. 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 SYSPARTS 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 1024KB. Only 240KB of storage was defined leaving 784KB of unmapped storage.

Changing the SYSPARTS Statement for Extended Address Mode:

You can specify up to 32 partitions for the following operands of the SYSPARTS statement for processors using extended address mode (4956 models E, 60E, H, J, and K). For example:

```
SYSPARTS NUMPART=32,                                                    X
         PARTS=(32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32),       X
         MAXPROG=(5,5,5,5,5,5,5,5,5,5,5,5,5,5,5,5)
```

The example above defines a 32-partition system with 5 concurrent programs in each of the first 16 partitions, and 10 concurrent programs in the remaining 16 partitions. The system defines each partition as 64K bytes in size.

SYSPARMS Statement

The SYSPARMS system partition statement defines the sizes of various system buffers and the partition in which to load certain system programs. If the SYSPARMS statement is coded, it must be specified only once and must follow the SYSPARTS statement and precede the SYSCOMM and SYSEND statements.

```
SYSPARMS    DATEFMT=DDMMYY,                                             X
            IABUF=40,                                                   X
            INITMOD=(CSXINIT,USERINIT),                                 X
            INITPRT=16,                                                 X
            LOGPART=20,                                                 X
            MECBLST=12,                                                 X
            TBPART=20,                                                  X
            VIRPART=20,                                                 X
            XPSSTK=30
```

The example above defines a system with the date in the day/month/year format. There are entry points for modules executed during system initialization. $INITIAL will be loaded in partition 16. $LOG and $VIRLOG will be loaded in partition 20. The unmapped storage table will be constructed in partition 20. The cross partition stack will contain 30 entries. The system will include 12 multiple event control blocks. The immediate action buffer (IABUF) will contain 40 entries.

See "SYSPARMS - Define Buffers and Partition" on page A-25 for a detailed description of this statement and the operands.

SYSCOMM Statement

The SYSCOMM system partition statement defines which partition will be used as the base partition and which partitions will contain common storage and the amount of common storage per partition (in 2K blocks). The SYSCOMM statement is optional, If specified, it must be specified only once and, it must follow the SYSPARTS and SYSPARMS statements and precede the SYSEND statement.

```
SYSCOMM     COMMON=CSXEND,                          X
            COMBASE=2
```

The previous example defines COMMON based in partition 2. Every partition defined by the NUMPART operand on the SYSPARTS statement has enough COMMON storage to support Communications Facility instructions.

See "SYSCOMM - Define Base and Common Partition" on page A-28 for a detailed description of this statement and the operands.

SYSEND Statement

The SYSEND system partition statement defines the end of the system partition statements area. The SYSEND statement is required and must be specified only once.

```
SYSEND
```

The previous example defines the end of the system partition statements area.

See "SYSEND - Define End of System Partition Statements" on page A-30 for a detailed description of this statement.

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 A-44 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 at the end of this chapter 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 A-10 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 A-117 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.

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 A-2 for a detailed description of this statement.

SYSMSG Statement

You can code the SYSMSG statement in the $EDXDEF data set to direct all system messages to a disk data set, to your $SYSLOG terminal, to the Communications Facility log, or to any combination of these destinations. See "SYSMSG - Define System Message Destination" on page A-21 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 A-46 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 A-46 for a detailed description of this statement.)

Notes:

1. The 3101 terminal (or equivalent device) 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 hardware or software 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, "Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals."
2. See "ACCA-Type Terminals" on page A-89 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 terminal,
- A 4978 terminal.

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
```

Changing the TERMINAL Statement for 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 4-3; the supervisor modules you must include are discussed in "Translation Table Support" on page 4-21 and "Terminal Support" on page 4-24. A detailed description of the Remote Support Link is presented in the *Problem Determination Guide* and the *Software Service Guide*.

Changing the TERMINAL Statement for Extended Address Mode:

With extended address mode, the partition with which a terminal is normally associated can be from 1 to 32. Therefore, the partition number in the PART= operand can be from 1 to 32 for the following devices:

- 2741
- 4013
- 4978
- 4979
- 4980
- ACCA
- TTY
- GPIB

The following example defines the device as a 4979 at address X'04' Partition 16 is the partition with which the terminal is associated.

```
TERM04    TERMINAL    DEVICE=4979,ADDRESS=04,PART=16
```

Changing the TERMINAL Statement for $VIRLOG Message Logging:

The $VIRLOG initialization option allows you to direct all messages sent to $SYSLOG to a disk data set, your $SYSLOG terminal, the Communications Facility log, or any combination of these. To define $VIRLOG message logging support, you must include the IOSVIRT supervisor module in the $LNKCNTL data set and include the proper TERMINAL and SYSMSG statements in the $EDXDEF data set. See "Interprogram Communication - Virtual Terminals" on page A-111 for information on coding the TERMINAL statement for $VIRLOG. See "SYSMSG - Define System Message Destination" on page A-21 for information on coding the SYSMSG statement for $VIRLOG.

The SMLLOAD initialization routine is automatically included during supervisor link time. If you code SYSMSG as DISK=YES and/or CF=YES, SMLLOAD loads $VIRLOG automatically when you IPL. All messages will be sent to the EDXSMLDS data set (on the IPL volume). If EDXSMLDS does not exist, the system allocates 200 records automatically if enough space is available.

BSCLINE Statement

The BSCLINE statement defines a binary synchronous communications line. Each line requires one BSCLINE statement to which programs that use the Binary Synchronous Communications Access Method (BSCAM) can refer.

See "BSCLINE - Define a Binary Synchronous Communications Line" on page A-6 for a detailed description of this statement.

LCC Statement

The LCC statement defines a Local Communications Controller. Each LCC attachment requires three LCC statements to which programs that use the Local Communication Controller Access Method can refer. With the LCC statement, you define all three subchannels of the attachment to EDX.

See "LCC - Define a Local Communication Controller Attachment" on page A-18 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 an SDLC line. For more information on defining an SDLC line, refer to the Program Directory for the Series/1 Systems Network Architecture, (5719-XX9).

See "EXIODEV - Define EXIO Interface Device" on page A-14 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 communications 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 Communications 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 A-17 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 A-19 for a detailed description of this statement.

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 if your COMBASE partition is partition 1. See "SYSCOMM - Define Base and Common Partition" on page A-28 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 at the end of this chapter, 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 5 - Defining Supervisor Structure" on page 4-33. 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.

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 4-31.

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.

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. Supervisor object modules that can reside in partitions 1 – 8 must precede EDXINIT in the $LNKCNTL data set if you move them to 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. One correct way to specify the partition would be 1, 1, 3, 3, 3, 2, 4. 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
*                     - MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------
PART 3
*SUPVIO              *28*  MAKE SUPERVISOR AND COMMON AREA STATIC
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
*SUPVIO              *28*  MAKE SUPERVISOR AND COMMON AREA STATIC
INCLUDE BSCAM        *7*   BISYNC COMMUNICATION SUPPORT
*-------------------------------------------------------------
```

VOLUME Statement

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

XS6006 XS6006 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 4-28.

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 (listed in the order the module appears in the $LNKCNTL data set).

Supervisor Support

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

EDXSYS EDXSYS is a required module. You must include it 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. You must include it 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 using it as it exists.

EDXSVCX EDXSVCX is a required module. You must include it in every supervisor. It performs task requests. For extended address mode systems, all modules included before EDXSVCX are mapped as static. (Refer to the *Customization Guide* for an explanation of static and dynamic partitions.)

$DBUGNUC $DBUGNUC must be included if you wish to debug programs using the $DEBUG utility.

EDXSTART EDXSTART is a required module. You must include it 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. You must include it if you want to use the wait-on-multiple-events option. If you include SWAITM, be sure to code the MECBLIST operand on the SYSPARMS system partition statement.

The following module can be included in any of the first eight supervisor partitions.

SUPVIO SUPVIO is an optional module, valid only with extended address mode support. You can include it in any supervisor partition to make the supervisor and COMMON area static. (Refer to the *Customization Guide* for more information.)

Timer Support

These supervisor object modules ***must*** reside 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. If you include EDXTIMER, do not include EDXTIMR2.

EDXTIMR2 EDXTIMR2 can 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. If you include EDXTIMR2, do not include EDXTIMER.

EXIO Control Support

These supervisor object modules ***must*** reside 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.

Optional Function Support

These supervisor object modules ***must*** reside 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 and LOADINT2. $XPSLINK also automatically includes LOADBUFR in a static area of partition 1.

The location of $LOADER on the disk is stored in the supervisor at IPL time. The first location where the supervisor looks for $LOADER is on the fixed-head volume. Then if your system does not have a fixed-head volume, the supervisor looks on the IPL volume. (If $LOADER is ever replaced, make sure you copy the new version to ***both*** the fixed-head volume and the IPL volume.)

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 4-21 for additional information).

STORMGR STORMGR is required to gain access to unmapped storage from an application program or to use the $MEMDISK utility to set up a memory disk volume. The $MEMDISK utility allows you to use part or all of unmapped storage as if it were a multivolume disk. Refer to the *Operator Commands and Utilities Reference* for information on the $MEMDISK utility. If you include STORMGR, $XPSLINK automatically includes STORINIT.

If you do not include STORMGR, $XPSLINK automatically includes NOSTRMGR. 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 you have or will install the Indexed Access Method Version 2 (5719-AM4).

PWRAM80 PWRAM80 is an optional module that you can include for the 4980 terminal. PWRAM80 automatically loads the control store, image store, and microcode necessary for 4980 terminal operation. If you do not include PWRAM80 and you turn off a 4980 terminal, you must IPL your system to make the 4980 operational again.

Host Communications Facility (HCF) Support

This supervisor object module ***must*** reside in partition 1.

TPCOM TPCOM is required if you defined host communications support (communication to a S/370) with the HOSTCOMM definition statement. If you include TPCOM, $XPSLINK automatically includes the initialization module TPINIT and also includes TPCOM1 in a static area of partition 1.

Translation Table Support

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

TRASCII TRASCII is required if you use feature numbers #1310, #2095/2096, and #7850 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 the *Problem Determination Guide* for more information about the Remote Support Link.

TREBASC TREBASC is required if you use feature numbers #1610 and #2091/2092 to connect ACCA or TTY I/O devices to your Series/1.

Note: You must include this module or TRASCII as well as both IOSTERM and IOSACCA for the Remote Support Link. Refer to the *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).

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. You must include IO1024 before EDXINIT.

System Support - Initialization

This module ***must*** reside in partition 1.

EDXINIT EDXINIT is a required module and must be included in every supervisor. It must appear in this location in $LNKCNTL.

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, refer to the *Customization Guide*. Also, supervisor object modules that can reside in partitions 1 – 8 must precede EDXINIT if you include them in partition 1.

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, refer to 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. Refer to the $XPSLINK 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.

EDXALU EDXALU is a required module. It is the Series/1 EDL interpreter. You can include it in any of the first eight partitions. If you move supervisor modules that contain emulation support for specific EDL instructions out of partition 1 to another partition, we recommend that you move them into the same partition as EDXALU to improve performance.

Communication Products Support

DLCROUTE must be included in the same partition as EDXALU. $SNASTUB may be included in partition 1 to 8.

DLCROUTE DLCROUTE is optional and is required only if you use Systems Network Architecture (SNA) with shared SDLC support, Primary Systems Network Architecture (PSNA), or Advanced Program-to-Program Communications (APPC). This module is the data link control router support for these products.

$SNASTUB $SNASTUB is optional and is required only if you use PSNA or APPC EDL instruction support.

Extended Address Mode Support

This supervisor object module can reside in any partition, 1 – 8.

SRMGR SRMGR is used for a 4956 models E, 60E, H, J, and K. It enables extended address mode support (more than 8 partitions).

Disk(ette) Support

These supervisor object modules can reside in any partition, 1 – 8. 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, $XPSLINK automatically includes $DISKIO.

DIDSKA DIDSKA must be included if you defined an IDSK disk using the DISK = IDSK definition statement.

DISKIOX DISKIOX must be included if you intend to use the dynamic data set extent function. Dynamic data set extent support provides automatic disk storage allocation when you exceed the initial data set size of an extendable data set, provided volume directory and disk space exist. If you include DISKIOX, DSKXINIT and DMEXTBL 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 you include D1024, the system automatically loads a copy of $IO1024 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, $XPSLINK automatically includes the initialization module TAPEINIT.

Terminal Support

These supervisor object modules can reside in any partition, 1 – 8. 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, 4224, 4201, 4202, 4978, 4979, 4980, 5219, 5224, 5225, 5262, 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 the system issues only the message text, not the parameters. If system commands and/or utilities are used, the messages will not contain the parameters and may not be meaningful. For systems with secondary storage (disk or diskette) or 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 or its equivalent in block mode (DEVICE = ACCA,MODE = 3101B). If you include this module, you must include IOSACCA as well.

IOS316X IOS316X is required if you are using a 3151, 3161, 3163, or 3164 in block mode (DEVICE = ACCA). If you include this module, you must include IOSACCA as well.

IOS4974 IOS4974 is optional and is required only if you defined a 4234, 4973, 4974, 4975 (except 4975-01A), 5219, 5262, 5224, and/or 5225 printer using the TERMINAL definition statement.

IOS4224 IOS4224 is optional. Include it only if you defined a 4201, 4202, or 4224 printer using the TERMINAL definition statement.

IOS4975A IOS4975A is optional. Include it only if you defined a 4975-01A, 4201, 4202, or 4224 printer using the TERMINAL definition statement. If you include this module, $XPSLINK automatically includes IOSTERM, IOSACCA, and $IO4975A.

IOS4979 IOS4979 is optional and is required only if you defined a 4978, 4979, and/or 4980 terminal using the TERMINAL definition statement. If you included the OPTION NOVERLAY statement, you must include INIT4978 for the 4978 and 4979 terminals and INIT4980 for the 4980 terminal.

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 Table Support" on page 4-21).

Note: You must include this module for the Remote Support Link. Refer to the *Problem Determination Guide* for more information about the Remote Support Link.

IOSHARE IOSHARE is optional and is required only if you are sharing a single physical line for two asynchronous devices. Refer to the *Communications Guide* for more information on line sharing.

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.
- Serial printer (4201, 4202, 4224)

Note: You must include this module for the Remote Support Link. Refer to the *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. You must include IOSVIRT if you intend to use the Remote Management Utility with the PASSTHRU function. You must also include IOSVIRT if you intend to use the $VIRLOG initialization option.

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 reside in any partition, 1–8.

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 supervisor object module can reside in any partition, 1–8.

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

Binary Synchronous Communications Support

These supervisor object modules can reside in any partition, 1–8.

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 Input/Output Support

These supervisor object modules can reside in any partition, 1 – 8. However, if they are left in partition 1, performance improves.

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.

Local Communications Controller (LCC) Support

This supervisor object module can reside in any partition, 1 – 8.

LCCAM You must include LCCAM if you defined a Local Communications Controller attachment with the LCC definition statement. If you include LCCAM, $XPSLINK automatically includes the initialization modules LCCINT, LCCPTC, LCPOV1, and LCPOV2. Refer to the *Communications Guide* for information on how to use LCC.

Additional Modules for NOVERLAY Option

If you include option NOVERLAY in your tailored system generation, the following initialization modules will ***not*** be included automatically by $XPSLINK. If you are using any of the devices they initialize, you must include these modules in partition 1 following EDXINIT.

RW4963ID RW4963ID is required to initialize a 4963 disk with fixed-head support 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 definition statement. Include this module only if you specify OPTION NOVERLAY. If you do, OPENADAP is included automatically.

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. If you do, OPENMFA is included automatically.

INIT4978 INIT4978 is required to initialize a 4978 or 4979 terminal defined with the TERMINAL definition statement. Include this module only if you specify OPTION NOVERLAY.

INIT4980 INIT4980 is required to initialize a 4980 terminal defined with the TERMINAL definition statement. Include this module only if you specify OPTION NOVERLAY. If you do, OPENADAP is included automatically.

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 alphanumeric 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

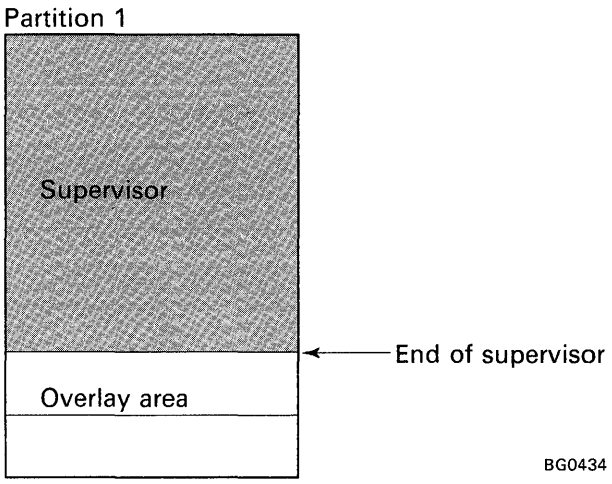
An overlay area is an area in storage that is used to execute overlay segments. There are two ways to create an overlay area in your operating system:

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
*
*-------------------------------------------------------------
*SUPPORT TO BE INCLUDED IN PARTITION 1 GOES AFTER HERE
*-------------------------------------------------------------
 PART1
*-------------------------------------------------------------
*SUPERVISOR SUPPORT       - MUST BE FIRST AND IN PARTITION 1
*-------------------------------------------------------------
 VOLUME XS6006                DEFAULT VOLUME FOR INCLUDE MODULES
*OVLAREA OVLSTART OVLEND *23* USER-DEFINED OVERLAY AREA
     .
     .
     .
*************************************************************
* SYSTEM SUPPORT -- INITIALIZATION - MUST BE IN PARTITION 1
*************************************************************
```

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



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:

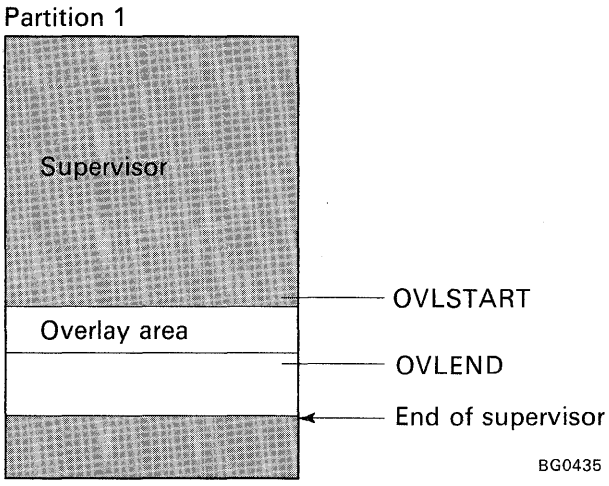
```
OVERLAY AREA IS TOO SMALL, SIZE REQUIRED IS xxx HEX
```

   where xxx indicates the required size of the overlay area in bytes (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
*
*-----------------------------------------------------------------
*SUPPORT TO BE INCLUDED IN PARTITION 1 GOES AFTER HERE
*-----------------------------------------------------------------
 PART1
*-----------------------------------------------------------------
*SUPERVISOR SUPPORT       - MUST BE FIRST AND IN PARTITION 1
*-----------------------------------------------------------------
 VOLUME XS6006                DEFAULT VOLUME FOR INCLUDE MODULES
 OVLAREA OVLSTART OVLEND *23* USER DEFINED OVERLAY AREA
     •
     •
     •
*****************************************************************
* SYSTEM SUPPORT -- INITIALIZATION - MUST BE IN PARTITION 1
*****************************************************************
```

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 and will default to an overlay structure unless you specify no 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.

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
*
*-------------------------------------------------------------
*SUPPORT TO BE INCLUDED IN PARTITION 1 GOES AFTER HERE
*-------------------------------------------------------------
 PART1
*-------------------------------------------------------------
*SUPERVISOR SUPPORT       -   MUST BE FIRST AND IN PARTITION 1
*-------------------------------------------------------------
 VOLUME XS6006                DEFAULT VOLUME FOR INCLUDE MODULES
*OVLAREA OVLSTART OVLEND *23* USER DEFINED OVERLAY AREA
     •
     •
     •
*************************************************************
* SYSTEM SUPPORT -- INITIALIZATION - MUST BE IN PARTITION 1
*************************************************************
 INCLUDE EDXINIT         *24* SUPERVISOR INITIALIZATION
 INCLUDE INITMFA          *3*  MFA INITIALIZATION
 INCLUDE INITADAP         *3*  ALPA & SMIO INITIALIZATION
 INCLUDE INIT4978         *3*  4978 DISPLAY INITIALIZATION
 INCLUDE INIT4980         *3*  4980 DISPLAY INITIALIZATION
 INCLUDE RW4963ID         *3*  4963 FIXED-HEAD REFRESH SUPPORT
 INCLUDE DSKINIT2         *3*  FOR DISK OR DISKETTE DEVICE
*
*************************************************************
```

Initialization Modules

Refer to the *Internal Design* for a description of the initialization modules that are automatically included in your supervisor, if required, by the linkage editor.

Step 4 - Estimate the Size of the Dynamic Data Set Extents Table (DMEXTBL)

If you did not include the DISKIOX supervisor module for dynamic data set extent support, skip to "Step 5 - Defining Supervisor Structure."

DMEXTBL is a table the system uses to hold information about dynamic data set extents. It consists of control information followed by a data area used by DISKIOX. This data area must be at least large enough to hold the information for the data set containing the largest number of extents. However, for best performance, the table should be large enough for all extent data sets that are open at the same time. Use the following formulas to estimate the size you need for your system:

```
( # of extent data sets open simultaneously) * 46 =
     size of primary area in the table (bytes)


(total # of extents for all data sets open simultaneously) * 20 =
     size of extent area in the table (bytes)
```

The DMEXTBL data set as shipped from IBM is large enough for 20 extended data sets and 100 extents. If you want to increase or decrease the table size, you need to edit the source module named DMEXTBL2. The procedure for editing this module is in "Step 5 - Edit DMEXTBL2 (Optional)" on page 5-27. Turn to that page now and write the size of the data table you just calculated. You will need this number when you perform your system generation.

Step 5 - Defining Supervisor Structure

In "Step 1 - Estimating Total Supervisor Size" on page 4-3, you estimated the total size of your supervisor. If your supervisor size is 64K or greater or you want to free 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 use less storage 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 a partition that does not contain 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 SYSPARTS system partition statement, you define the partition structure and the amount of storage assigned to each partition with the NUMPART= and PARTS= operands. 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 1. 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.

Part B - Estimate Storage Needed by Other Programs

The supervisor requires storage 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 each program requires. 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 SYSPARTS statement.

Work Sheet 3 The PART statement in $LNKCNTL indicating which modules are being moved from partition 1 and the partition where they will be located.

Work Sheets for the Sample System

This section contains 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 4-36 shows the definition statements defining the devices and "Work Sheet 3 for Sample System" on page 4-39 shows the software support selected.

Work Sheet 2 for Sample System

System definition statements				Required or Optional
Storage definition				
blank	SYSPARTS	NUMPART=	7,	Required
		PARTS=	(0,24,16,20, 32,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___),	Optional
		MAXPROG=	(1,10,10,5, 5,___,___,___ ___,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___),	Optional
blank	SYSPARMS	DATEFMT=	_____,	Optional
		IABUF=	_____,	Optional
		INITMOD=	___,___,___,___,	Optional
		INITPRT=	_____,	Optional
		LOGPART=	_____,	Optional
		MECBLST=	_____,	Optional
		TBPART=	_____,	Optional
		VIRPART=	_____,	Optional
		XPSSTK=	_____,	Optional
blank	SYSCOMM	COMMON=	(0,2,0,1, 1,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___, ___,___,___,___),	Required

Figure 4-2 (Part 1 of 3). Work Sheet 2 for the Sample System

System definition statements				Required or Optional
		COMBASE=	_____,	Optional
			_____,	
			_____,	
			_____,	
			_____,	
			_____,	
			_____,	
			_____,	
blank	SYSEND			Required
			.	
			.	
			.	
Disk(ette) definition				
blank	DISK	DEVICE=	**4962-1F,**	Required
		ADDRESS=	**03**,	Required
		VOLNAME=	(_____),	Optional
		TASK=	**YES**,	Optional
		END=	(_____)	Optional
			.	
			.	
			.	
blank	DISK	DEVICE=	**4964,**	Required
		ADDRESS=	**02**,	Required
		VOLNAME=	(_____),	Optional
		TASK=	**YES**,	Optional
		END=	**YES**	Optional
			.	
			.	
			.	
4973/4974 Printers				
$SYSPRTR	TERMINAL	DEVICE=	**4973,**	Required
		ADDRESS=	**21**,	Required
		PAGSIZE=	_____,	Optional
		LINSIZE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		SPOOL=	_____,	Optional
		END=	_____	Optional
			.	
			.	
			.	

Figure 4-2 (Part 2 of 3). Work Sheet 2 for the Sample System

System definition statements				Required or Optional
4978/4979 Terminals				
$SYSLOG	TERMINAL	DEVICE=	**4979,**	Required
		ADDRESS=	**04**,	Required
		LINSIZE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		NHIST=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		HDCOPY=	**$SYSPRTR,**	Optional
		ATTN=	_____,	Optional
		PF1=	_____,	Optional
		SCREEN=	_____,	Optional
		PART=	_____,	Optional
		SPOOL=	_____,	Optional
		END=	_____	Optional
Second Terminal				
_____				
$SYSLOGA	TERMINAL	DEVICE=	**4978,**	
		ADDRESS=	**24**,	
		LINSIZE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		NHIST=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		HDCOPY=	**$SYSPRTR,**	
		ATTN=	_____,	Optional
		PF1=	_____,	Optional
		SCREEN=	_____,	Optional
		PART=	_____,	Optional
		SPOOL=	_____,	Optional
		END=	**YES**	
			.	
			.	
			.	
System Common				
	$SYSCOM	CSECT		
		QCB		
		QCB	NONE DEFINED	
		ECB		
		ECB		
		ENTRY	$EDXPTCH	
	$EDXPTCH	DATA	128F'0'	System Patch Area

Figure 4-2 (Part 3 of 3). Work Sheet 2 for the Sample System

Work Sheet 3 for Sample System

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
	PART 1		
*	Supervisor support	-	must be first and in partition 1
	VOLUME XS6006		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 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
*	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
*	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

Figure 4-3 (Part 1 of 4). Work Sheet 3 for the Sample System

Column One	Supervisor Object Modules	Notes	Purpose of Module
*	INCLUDE TREBCD	*11*	2741, PROC EBCD translation
*	INCLUDE TRCRSP	*11*	2741 correspondence translation
*	System initialization	-	must be in partition 1
*	INCLUDE $PROG1	*22*	User module included in nucleus gen
*	INCLUDE IO1024	*21*	1024 IPL support
*	SNA EDL stub module		
*	INCLUDE $SNASTUB	*33*	SNA EDL instruction stub module
*	System support - initialization	-	must be in partition 1
	INCLUDE EDXINIT	*24*	Supervisor initialization
	PART 2		
*	INCLUDE SUPVIO	*28*	Make supervisor and common area static
	INCLUDE EDXALU	*30*	EDL instruction emulator
*	INCLUDE DLCROUTE	*34*	Data link control router
*	INCLUDE CDXQAPND	*34*	Data link control router sub module
*	INCLUDE CDXQPULL	*34*	Data link control router sub module
*	INCLUDE CDXQWAIT	*34*	Data link control router sub module
*	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
	INCLUDE DISKIO	*3*	Basic disk(ette) support
*	INCLUDE DISKIOX	*31*	Dynamic data set extent support - optional
	INCLUDE D49624	*3*	4962/4964 disk(ette) support
*	INCLUDE D4963A	*3*	4963/4967/DDSK disk support
*	INCLUDE D4966A	*3*	4965/4966 diskette support
*	INCLUDE DIDSKA	*3*	IDSK disk(ette) 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
	INCLUDE EDXTIO	*4*	Basic terminal support
*	INCLUDE MINMSG	*5*	Minimum message support
	INCLUDE FULLMSG	*5*	Full message support
*	INCLUDE IOS3101	*7*	ACCA 3101B support

Figure 4-3 (Part 2 of 4). Work Sheet 3 for the Sample System

Column One	Supervisor Object Modules	Notes	Purpose of Module
*	INCLUDE IOS316X	*7*	ACCA 3161B/3163B/3164B support (3151)
	INCLUDE IOS4979	*3*	4978/4979/4980 display support
	INCLUDE IOS4974	*3*	4234/4973/4974/4975/5219 /5224/5225/5262 support
*	INCLUDE IOS4224	*3*	4201/4202/4224 TERMCTRL support
*	INCLUDE IOS4975A	*3*	ACCA 4975-01A/4224/4201 support
*	INCLUDE IOSACCA	*8*	ACCA device handler
*	INCLUDE ACCATRC	*3*	ACCA trace option
*	INCLUDE IOSHARE	*3*	ACCA line sharing support
*	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 included in partition 1 to 8
*	INCLUDE EDXFLOAT	*3*	Floating point arithmetic
*	Queue I/O	-	May be included in partition 1 to 8
*	INCLUDE QUEUEIO	*19*	Queue processing support
*	Bisync communications	-	Must be grouped together May be included in partition 1 to 8
*	INCLUDE BSCAM	*13*	Bisync communication
*	INCLUDE BSCX21	*27*	Bisync communication support for X.21
*	Sensor input/output	-	Must be grouped together May be included in partition 1 to 8
*	INCLUDE SBCOM	*15*	Basic sensor I/O support
*	INCLUDE SBAI	*3*	Analog input support
*	INCLUDE SBAO	*3*	Analog output support
*	INCLUDE SBDIDO	*3*	Digital I/O support
*	INCLUDE SBPI	*3*	Process interrupt support
*	LCC communications	-	May be included in partition 1 to 8
*	INCLUDE LCCAM	*3*	Local Communications Controller support

Figure 4-3 (Part 3 of 4). Work Sheet 3 for the Sample System

Column One	Supervisor Object Modules	Notes	Purpose of Module
*	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
	Insert user initialization modules here		NOT REQUIRED
	Location of supervisor		
	LINK **$EDXNUC2**,EDX002		REPLACE END

Figure 4-3 (Part 4 of 4). Work Sheet 3 for the Sample System

Chapter 5. Generate a Tailored Operating System

This chapter provides the step-by-step procedures for generating a tailored operating system. Before you generate an operating system, you should have completed work sheet 2 and work sheet 3. Work sheet 2 contains the system partition 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 a 4952, 4954, 4955, or 4956 Processor."

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 ensure the system definition statements 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 DMEXTBL2 on the IPL volume to change the size of the dynamic data set extents table.

Step 6 Edit SCBTBL on volume ASMLIB to increase or decrease the size of the segmentation register control block table.

Step 7 Edit the $JOBUTIL procedure file ($SUPPREP on volume ASMLIB) to use the data sets allocated in Step 2.

Step 8 Execute $SUPPREP, using the $JOBUTIL utility and the job stream procedure created in Step 7, 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 (EDXNUCT, for example) on EDX002.

Step 9 Edit $SRPROF on volume ASMLIB to specify the IPL configuration profile (extended address mode only).

Step 10 Initialize your tailored operating system.

Step 11 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 (loaded by the procedure data set).

$XPSLINK $XPSLINK is used to link edit and format the supervisor (loaded 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 a 4952, 4954, 4955, or 4956 Processor," 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: "Step 8 - Exercise the Utilities and Program Preparation Facilities" on page 3-20 suggests that you assemble the sample program CALCSRC to verify starter system 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 5-1. 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. Refer to $DISKUT1 in the *Operator Commands and Utilities Reference* or the *Operation Guide* for an explanation of organization types.

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

```
      > $L $DISKUT1
      LOADING $DISKUT1     nnP, LP=nnnn, PART=x
      $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. See "Work Sheet 2" on page E-5 for this work sheet. 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    nnP, LP=nnnn, PART=x

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 **Y** and continue.

```
$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
```

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

1. Enter a **3** on the OPTION 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 **ASMLIB** 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:

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

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

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

```
$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 5-2 on page 5-6 shows the contents of $EDXDEF.

```
$EDXDEF  CSECT
         $ID
*                      XPS
*                EVENT DRIVEN EXECUTIVE
*             VERSION 6, MODIFICATION LEVEL 0
*
* THE FOLLOWING DEFINES THE STARTER SUPERVISOR AS SHIPPED ON THE
* DISKETTE LABELED XS6006.  FOR COMPLETE DESCRIPTIONS OF THESE
* STATEMENTS OR ANY OTHER SYSTEM DEFINITION STATEMENTS, REFER TO
* THE INSTALLATION AND SYSTEM GENERATION GUIDE, SC34-0936.
*
* ONLY STATEMENTS WITHOUT AN ASTERISK (*) IN COLUMN 1 ARE ASSEMBLED.
*
*--------------------------------------------------------------------
* BASIC STORAGE DEFINITIONS
*--------------------------------------------------------------------
* PARTITION NUMBER:      1  2  3  4  5  6  7  8
    SYSPARTS NUMPART=8,                                              C
              PARTS=(32,32,32,32,32,32,32,32),                       C
              MAXPROG=(05,05,05,05,05,05,05,05)
*
* PARTITION #:  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
*               17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
*   SYSPARTS NUMPART=32,                                           C
*             PARTS=                                               C
*             (32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32     C
*             ,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32),   C
*             MAXPROG=                                             C
*             (05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05     C
*             ,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05,05)
*--------------------------------------------------------------------
* SYSTEM PARAMETERS
*--------------------------------------------------------------------
    SYSPARMS DATEFMT=MMDDYY,                                         C
              IABUF=20,                                              C
              INITPRT=0,                                             C
              LOGPART=0,                                             C
              TBPART=0,                                              C
              VIRPART=0,                                             C
              XPSSTK=20
*--------------------------------------------------------------------
* SYSTEM COMMON BASE AND COMMON OPERANDS
*--------------------------------------------------------------------
*   SYSCOMM  COMMON=(),                                            C
*             COMBASE=1
*--------------------------------------------------------------------
* END OF SYSTEM STATEMENTS
*--------------------------------------------------------------------
    SYSEND
*--------------------------------------------------------------------
* 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
*--------------------------------------------------------------------
```

Figure 5-2 (Part 1 of 4). $EDXDEF Data Set

```
* BASIC DEFINITION FOR A 4963-23 DISK              (23 MB)
*-------------------------------------------------------------------
        DISK   DEVICE=4963-23,ADDRESS=48
*-------------------------------------------------------------------
* BASIC DEFINITION FOR A 4967-2 OR -4 DISK
*-------------------------------------------------------------------
*       DISK   DEVICE=4967-2,ADDRESS=48
*-------------------------------------------------------------------
* BASIC DEFINITION FOR A DDSK-30 DISK              (30 MB)
*-------------------------------------------------------------------
*       DISK   DEVICE=DDSK-30,ADDRESS=44
*-------------------------------------------------------------------
* BASIC DEFINITION FOR A DDSK-60 DISK              (60 MB)
*-------------------------------------------------------------------
*       DISK   DEVICE=DDSK-60,ADDRESS=44
*-------------------------------------------------------------------
* BASIC DEFINITION FOR AN IDSK DISK
*-------------------------------------------------------------------
*       DISK   DEVICE=IDSK,ADDRESS=60
*-------------------------------------------------------------------
*-------------------------------------------------------------------
* DISKETTE DEFINITIONS
*-------------------------------------------------------------------
        DISK   DEVICE=4964,ADDRESS=02
        DISK   DEVICE=4965,ADDRESS=44
        DISK   DEVICE=4965,ADDRESS=45
        DISK   DEVICE=IDSK,ADDRESS=61
        DISK   DEVICE=4966,ADDRESS=22,END=YES
*-------------------------------------------------------------------
* TIMER DEFINITION
*-------------------------------------------------------------------
*       TIMER  ADDRESS=40
*-------------------------------------------------------------------
* ADAPTER DEFINITIONS
*-------------------------------------------------------------------
SMIO04   ADAPTER   TYPE=SMIO,ADDRESS=80,                          C
             DEVICES=($SYSLOG)
MFA58    ADAPTER   TYPE=MFA,ADDRESS=58,                           C
              DEVICES=($SYSLOGB,MPRTR1,MPRTR2),END=YES
*ALPAC0  ADAPTER   TYPE=ALPA,ADDRESS=C0,                          C
*             DEVICES=(MPRTR3),END=YES
*-------------------------------------------------------------------
* BASIC DEFINITIONS FOR SYSTEM MESSAGE LOGGING
*-------------------------------------------------------------------
*        SYSMSG    TERM=YES,DISK=NO,CF=NO
*$$SYSLOG TERMINAL  DEVICE=VIRT,ADDRESS=X$SYSLOG
*X$SYSLOG TERMINAL  DEVICE=VIRT,ADDRESS=$$SYSLOG,SYNC=YES
*-------------------------------------------------------------------
```

Figure 5-2 (Part 2 of 4). $EDXDEF Data Set

```
* 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,BITRATE=9600
$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
*MPRTR0   TERMINAL DEVICE=4224,ADDRESS=58,CODTYPE=ASCII,               C
*              BITRATE=9600,ADAPTER=MFA
```

Figure 5-2 (Part 3 of 4). $EDXDEF Data Set

```
*MPRTRO   TERMINAL DEVICE=4201,ADDRESS=58,CODTYPE=ASCII,                C
*             BITRATE=4800,ADAPTER=MFA,LINSIZE=132,RIGHTM=131
*REMSUPT TERMINAL  DEVICE=ACCA,ADDRESS=08,BITRATE=1200,                 C
*             LMODE=SWITCHED,ADAPTER=SINGLE,PF1=D886,                   C
*             CODTYPE=EBASC,ATTN=D816,PAGSIZE=24,SCREEN=YES
*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
*-------------------------------------------------------------------
* BINARY SYNCHRONOUS COMMUNICATIONS LINE DEFINITION
*-------------------------------------------------------------------
*         BSCLINE ADDRESS=09,END=YES
*-------------------------------------------------------------------
* EXIO INTERFACE DEVICE DEFINITION
*-------------------------------------------------------------------
*         EXIODEV  ADDRESS=0A,MAXDCB=16,RSB=4,END=YES
*-------------------------------------------------------------------
* LOCAL COMMUNICATIONS CONTROLLER ATTACHMENT DEFINITION
*-------------------------------------------------------------------
*         LCC      ADDRESS=50
*         LCC      ADDRESS=51
*         LCC      ADDRESS=52,END=YES
*-------------------------------------------------------------------
* SYSTEM  COMMON
*-------------------------------------------------------------------
$SYSCOM  CSECT
         QCB
         QCB
         ECB
         ECB
         ENTRY    $EDXPTCH
$EDXPTCH DATA     128F'0'              SYSTEM PATCH AREA
         END
```

Figure 5-2 (Part 4 of 4). $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, refer to $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.

3(e) Define support to direct $SYSLOG messages to a disk data set.

In order to use the $VIRLOG initialization option to direct all your $SYSLOG messages to a disk data set, you must include the IOSVIRT module (in the $LNKCNTL data set) and edit $EDXDEF as follows:

```
$EDXDEF  CSECT
         $ID
*                        XPS
*                 EVENT DRIVEN EXECUTIVE
*              VERSION 6, MODIFICATION LEVEL 0
                         •
                         •
                         •
*****************************************************************
* BASIC DEFINITION FOR SYSTEM MESSAGE LOGGING
*****************************************************************
*
           SYSMSG    TERM=YES,DISK=YES,CF=NO
$$SYSLOG   TERMINAL  DEVICE=VIRT,ADDRESS=X$SYSLOG
X$SYSLOG   TERMINAL  DEVICE=VIRT,ADDRESS=$$SYSLOG,SYNC=YES
```

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

3(f) 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 6, MODIFICATION LEVEL 0
*
                         •
                         •
                         •
```

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

3(g) 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.

Example 1: The following example shows the definition data set as modified for the sample system:

```
$EDXDEF  CSECT
         $ID
*                        XPS
*                 EVENT DRIVEN EXECUTIVE
*              VERSION 6, MODIFICATION LEVEL 0
*
* THE FOLLOWING DEFINES THE STARTER SUPERVISOR AS SHIPPED ON THE
* DISKETTE LABELED XS6006.  FOR COMPLETE DESCRIPTIONS OF THESE
* STATEMENTS OR ANY OTHER SYSTEM DEFINITION STATEMENTS, REFER TO
* THE INSTALLATION AND SYSTEM GENERATION GUIDE, SC34-0936.
*
* 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
*                     17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
    SYSPARTS  NUMPART=5,                                             C
              PARTS=(0,24,16,20,32),                                 C
              MAXPROG=(1,10,10,5,5)
*-------------------------------------------------------------------
* SYSTEM PARAMETERS
*-------------------------------------------------------------------
    SYSPARMS DATEFMT=MMDDYY,                                         C
              IABUF=20,                                              C
              INITPRT=0,                                             C
              LOGPART=0,                                             C
              TBPART=0,                                              C
              VIRPART=0,                                             C
              XPSSTK=20
*-------------------------------------------------------------------
* SYSTEM COMMON BASE AND COMMON OPERANDS
*-------------------------------------------------------------------
    SYSCOMM  COMMON=(0,2,0,1,1),                                     C
              COMBASE=1
*-------------------------------------------------------------------
* END OF SYSTEM STATEMENTS
*-------------------------------------------------------------------
    SYSEND
*-------------------------------------------------------------------
*-------------------------------------------------------------------
* 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
*-------------------------------------------------------------------
```

Figure 5-3 (Part 1 of 2). Edited $EDXDEFx Data Set

```
* TERMINAL DEFINITIONS
*-------------------------------------------------------------------
$SYSPRTR TERMINAL DEVICE=4973,ADDRESS=21
$SYSLOG  TERMINAL DEVICE=4979,ADDRESS=04,HDCOPY=$SYSPRTR
$SYSLOGA TERMINAL DEVICE=4978,ADDRESS=24,HDCOPY=$SYSPRTR,END=YES
*-------------------------------------------------------------------
* SYSTEM COMMON
*-------------------------------------------------------------------
$SYSCOM  CSECT
*
* NONE DEFINED
*
         ENTRY    $EDXPTCH
$EDXPTCH DATA     128F'0'       SYSTEM PATCH AREA
         END
```

Figure 5-3 (Part 2 of 2). Edited $EDXDEFx Data Set

3(h) Enter MENU on the COMMAND INPUT line to return to the primary option menu. Then select option **8** and press the enter key to end $FSEDIT.

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. Statements for 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) 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    nnP, LP=nnnn, PART=x
```

Once loaded, $FSEDIT displays the primary option menu.

4(b) Read a copy of the link-control data set into the work data set.

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.

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

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

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

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

where *nn* is the number of lines in the data set.

4(c) Enter a **2** (EDIT) on the OPTION LINE to edit $LNKCNTL.

```
$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 contents of the data set. The following figure shows the contents of $LNKCNTL.

```
*                    STARTER SYSTEM
*                 EVENT DRIVEN EXECUTIVE
*           XPS - VERSION 6, 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
*
************************************************************************
*  SUPPORT TO BE INCLUDED IN PARTITION 1 GOES AFTER HERE
************************************************************************
 PART 1
*----------------------------------------------------------------------
*  SUPERVISOR SUPPORT     - MUST BE FIRST AND IN PARTITION 1
*----------------------------------------------------------------------
 VOLUME  XS6006                  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 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
```

Figure 5-4 (Part 1 of 7). $LNKCNTL Data Set Contents

```
*-------------------------------------------------------------------
*  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
*
*-------------------------------------------------------------------
*  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
*
*-------------------------------------------------------------------
*  SYSTEM INITIALIZATION - MUST BE IN PARTITION 1
*-------------------------------------------------------------------
*INCLUDE $PROG1           *22*   USER MODULE INCLUDED IN NUCLEUS GEN
*INCLUDE IO1024           *21*   1024 IPL SUPPORT
*
*-------------------------------------------------------------------
*INCLUDE $SNASTUB         *33*   SNA EDL INSTRUCTION STUB MODULE
*-------------------------------------------------------------------
*  SYSTEM SUPPORT -- INITIALIZATION - MUST BE IN PARTITION 1
*-------------------------------------------------------------------
 INCLUDE EDXINIT          *24*   SUPERVISOR INITIALIZATION
*
*-------------------------------------------------------------------
*  INSERT USER INITIALIZATION MODULES HERE
*-------------------------------------------------------------------
*
*-------------------------------------------------------------------
*  SUPERVISOR CODE BEING MOVED OUT OF
*  PARTITION 1 MUST BE MOVED TO HERE
*-------------------------------------------------------------------
*
*********************************************************************
*  SUPPORT TO BE INCLUDED IN PARTITION 2 GOES AFTER HERE
*********************************************************************
 PART 2
*INCLUDE SUPVIO           *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
 INCLUDE EDXALU           *30*   EDL INSTRUCTION EMULATOR
*INCLUDE DLCROUTE         *34*   DATA LINK CONTROL ROUTER
*INCLUDE CDXQAPND         *34*   DATA LINK CONTROL ROUTER SUB MODULE
*INCLUDE CDXQPULL         *34*   DATA LINK CONTROL ROUTER SUB MODULE
*INCLUDE CDXQWAIT         *34*   DATA LINK CONTROL ROUTER SUB MODULE
```

Figure 5-4 (Part 2 of 7). $LNKCNTL Data Set Contents

```
*-------------------------------------------------------------------
*  EXTENDED ADDRESS MODE - MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
*INCLUDE SRMGR           *29*   EXTENDED ADDRESS MODE SUPPORT
*
*-------------------------------------------------------------------
*  DISK(ETTE) SUPPORT    - MUST BE GROUPED TOGETHER
*                        - MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
 INCLUDE DISKIO          *3*   BASIC DISK(ETTE) SUPPORT
*INCLUDE DISKIOX         *31*  DYNAMIC DATA SET EXTENT SUPPORT-OPTIONAL
 INCLUDE D49624          *3*   4962/4964 DISK(ETTE) SUPPORT
 INCLUDE D4963A          *3*   4963/4967/DDSK DISK SUPPORT
 INCLUDE D4966A          *3*   4965/4966 DISKETTE SUPPORT
 INCLUDE DIDSKA          *3*   IDSK DISK(ETTE) 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
*-------------------------------------------------------------------
 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 IOS316X         *32*  ACCA 3161B/3163B/3164B SUPPORT (3151)
 INCLUDE IOS4979         *3*   4978/4979/4980 DISPLAY SUPPORT
 INCLUDE IOS4974         *3*   4234/4973/4974/4975/5219/5224/5225/5262 SUPPORT
*INCLUDE IOS4224         *3*   4201/4202/4224 TERMCTRL SUPPORT
*INCLUDE IOS4975A        *3*   ACCA 4975-01A/4201/4202/4224 SUPPORT
 INCLUDE IOSACCA         *8*   ACCA DEVICE HANDLER
*INCLUDE ACCATRC         *3*   ACCA TRACE OPTION
 INCLUDE IOSTERM         *A,6* ACCA/TTY/2741/4013 SUPPORT
 INCLUDE IOSHARE         *3*   ACCA LINE SHARING 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.
*
*-------------------------------------------------------------------
*  FLOATING POINT        - MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
*INCLUDE EDXFLOAT         *3*   FLOATING POINT ARITHMETIC
*
*-------------------------------------------------------------------
```

Figure 5-4 (Part 3 of 7). $LNKCNTL Data Set Contents

```
*-------------------------------------------------------------------
*  QUEUE I/O             - MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
*INCLUDE QUEUEIO         *19*  QUEUE PROCESSING SUPPORT
*
*-------------------------------------------------------------------
*  BISYNC COMMUNICATIONS  - MUST BE GROUPED TOGETHER
*                         - MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
*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
*-------------------------------------------------------------------
*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
*
*-------------------------------------------------------------------
*  LCC COMMUNICATIONS     - MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
*INCLUDE LCCAM           *3*   LOCAL COMMUNICATIONS CONTROLLER SUPPORT
*
*********************************************************************
*  SUPPORT TO BE INCLUDED IN PARTITION 3 GOES AFTER HERE
*********************************************************************
*PART 3
*INCLUDE SUPVIO          *28*  MAKE SUPERVISOR AND COMMON AREA STATIC
*
*********************************************************************
*  SUPPORT TO BE INCLUDED IN PARTITION 4 GOES AFTER HERE
*********************************************************************
*PART 4
*INCLUDE SUPVIO          *28*  MAKE SUPERVISOR AND COMMON AREA STATIC
*
*********************************************************************
*  SUPPORT TO BE INCLUDED IN PARTITION 5 GOES AFTER HERE
*********************************************************************
*PART 5
*INCLUDE SUPVIO         *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*
*********************************************************************
```

Figure 5-4 (Part 4 of 7). $LNKCNTL Data Set Contents

```
*********************************************************************
*  SUPPORT TO BE INCLUDED IN PARTITION 6 GOES AFTER HERE
*********************************************************************
*PART 6
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*
*********************************************************************
*  SUPPORT TO BE INCLUDED IN PARTITION 7 GOES AFTER HERE
*********************************************************************
*PART 7
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*
*********************************************************************
*  SUPPORT TO BE INCLUDED IN PARTITION 8 GOES AFTER HERE
*********************************************************************
*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 4234,4973,4974,
*    4975, 5219, 5224, 5225, 5262, 4978, 4979, 4980, GPIB, S/1-S/1, ACCA,
*    TTY, 2741, 3101, 4013, 3151, 3161, 3163, 3164, 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-31XXB
     TERMINALS.  IOSTERM ALSO REQUIRED WITH IOS4224 OR IOS4975A.
*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.
*    IOSACCA IS ALSO REQUIRED IF DEVICE=4224 OR DEVICE=4201.
*    APPROPRIATE TRANSLATION TABLE(S) MUST ALSO BE INCLUDED.
*    SEE NOTES 6, 7, 10, 11.
*9*  REQUIRED IF USING REMOTE MANAGEMENT UTILITY WITH PASSTHRU FUNCTION
*    OR $VIRLOG SUPPORT.
*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.
```

Figure 5-4 (Part 5 of 7). $LNKCNTL Data Set Contents

```
*11* TREBCD AND/OR TRCRSP  ARE REQUIRED IF IOS2741 IS INCLUDED.
*    2741 TERMINALS MAY USE EITHER TREBCD OR TRCRSP.
*    DEVICE=PROC NORMALLY USES TREBCD.
*12* ATTACHED TIMERS (FEATURE 7840) AND THE 4952/4954/4956 NATIVE TIMERS
*    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 4980 TERMINALS, IN CASE OF
*    POWER ON/OFF, WITHOUT RE-IPLING THE SYSTEM.
```

Figure 5-4 (Part 6 of 7). $LNKCNTL Data Set Contents

```
*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 INSTALLATION
*    AND SYSTEM GENERATION GUIDE, SC34-0936.
*30* THIS MODULE MAY INCLUDED IN PARTITION 1 TO 8.  IF THIS MODULE
*    IS MOVED OUT OF PARTITION 1 (THE DEFAULT IS PARTITION 2), IT IS
*    RECOMMENDED THAT ALL OTHER SUPPORT THAT IS MOVABLE BE INCLUDED
*    IN THE SAME PARTITION.
*31* INCLUDE THIS MODULE FOR DATA SET EXTENTS SUPPORT.  THIS MODLUE
*    MUST BE IN THE SAME PARTITION AS THE MODULE DISKIO.
*32* IOS316X IS REQUIRED IF A 316X TERMINAL IS TO BE USED IN BLOCK MODE
*    (E.G. DEVICE=ACCA,MODE=3161B.  IOSACCA MUST ALSO BE INCLUDED.
*    SEE NOTES 6,8,10,11.
*33* $SNASTUB IS REQUIRED FOR THE SNA EDL INSTRUCTIONS.
     THIS MODULE MAY BE INCLUDED IN PARTITION 1 TO 8.
*34* DATA LINK CONTROL ROUTER.  MUST BE INCLUDED IN THE SAME PARTITION
     AS EDXALU.
*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 5-4 (Part 7 of 7). $LNKCNTL Data Set Contents

Note: Refer to the *Customization Guide* for information about SUPVIO.

4(d) Edit $LNKCNTL.

Use $FSEDIT to add to, modify, or delete from the contents of $LNKCNTL to match your required support. 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. Edit the link-control data set to match the work sheet, inserting asterisks to omit modules or removing asterisks to include the appropriate modules.

Notes:

1. You may wish to change the name of the supervisor data set to $EDXNUCx (x = any alphanumeric 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.
2. Instead of deleting statements you don't need, simply 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 outside of partition 1. If you are generating a single partition supervisor, you do not have to change the PART statements in the link-control data set. If you are generating a multipartition supervisor, you should have indicated on work sheet 3 the number of the partition where you want to locate a specific group of modules on. Move this group of modules in the $LNKCNTL data set from partition 1 to the partition you specified on the work sheet.

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 move the second group of modules to follow the 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 you precede each module or group of modules with an OVERLAY statement. In addition, you must precede the initialization modules with 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.

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

```
 OPTION NOVERLAY        *25* NO OVERLAY STRUCTURE
*-------------------------------------------------------------
* 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 INITIALIZATION
 INCLUDE INITADAP       *3*  ALPA & SMIO INITIALIZATION
 INCLUDE INIT4978       *3*  4978 TERMINAL INITIALIZATION
 INCLUDE INIT4980       *3*  4980 TERMINAL INITIALIZATION
*-------------------------------------------------------------
* INSERT USER INITIALIZATION MODULES HERE
*-------------------------------------------------------------
 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
*-------------------------------------------------------------
 PART 3
*-------------------------------------------------------------
* SENSOR INPUT/OUTPUT - MUST BE GROUPED TOGETHER
*                     - MAY BE INCLUDED IN PARTITION 1 to 8
*-------------------------------------------------------------
 INCLUDE SUPVIO         *28*  MAKE SUPERVISOR AND COMMON AREA STATIC
 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
*-------------------------------------------------------------
 PART 4
*-------------------------------------------------------------
* BISYNC COMMUNICATION - MUST BE GROUPED TOGETHER
*                      - MAY BE INCLUDED IN PARTITION 1 to 8
*-------------------------------------------------------------
 INCLUDE BSCAM          *7*   BISYNC COMMUNICATION SUPPORT
 INCLUDE SUPVIO         *28*  MAKE SUPERVISOR AND COMMON AREA STATIC
*-------------------------------------------------------------
```

Figure 5-5. 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.

4(e) Including SRMGR (optional)

You must include the object module SRMGR if you intend to use extended address mode (4956 models E, 60E, H, J, and K only). If you include SRMGR, the system automatically includes RLOADER and EDXTIMR2 (if you have not included EDXTIMER). If you do not include SRMGR when you generate your system, then the system will use the current 3-bit architecture (8 partitions) which does not support extended address mode (32 partitions).

You can include SRMGR in partitions 1 through 8.

The following is a partial listing of the $LNKCNTL data set showing the module that pertains to extended address mode:

```
 PART 3
 INCLUDE SUPVIO          *28*  MAKE SUPERVISOR AND COMMON AREA STATIC
*                          •
*                          •
*                          •
*-------------------------------------------------------------------
* EXTENDED ADDRESS MODE  -    MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
 INCLUDE SRMGR           *29*  INCLUDE FOR EXTENDED ADDRESS MODE SUPPORT
*                          •
*                          •
*                          •
```

Figure 5-6. Partial $LNKCNTL Data Set for Extended Address Mode

4(f) Enter MENU on the COMMAND INPUT line to end the EDIT mode as shown.

```
EDIT --- $LNKCNTL,ASMLIB   236 ( 543)------------------COLUMNS 001 072
COMMAND INPUT ===> MENU                                 SCROLL ===>HALF
***** ***** TOP OF DATA ************************************************
*
*
* EVENT DRIVEN EXECUTIVE - VERSION 6, 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,XS6006   *1*   SYSTEM TABLES AND WORK AREA
```

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

4(g) Save the changed data set.

You may want to maintain more than one supervisor. For example, you may need one system for program development and one for production. We recommend 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 LINKCNTx, where *x* is any alphanumeric character ("L" in the following examples).

1. Enter a **4** on the OPTION line.
2. Enter the data set name and volume 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 ==========> LINKCNTL              (CURRENTLY IN WORK FILE)
VOLUME NAME ===========> EDX002
          •
          •
          •

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

$FSEDIT saves the contents of the work data set in LINKCNTL and 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  XS6006                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 EDXSTART         *1*   INITIALIZATION & ERROR HANDLER
*INCLUDE SWAITM           *3*   WAIT ON MULTIPLE EVENTS
*-------------------------------------------------------------------
*  EDX EMULATOR SUPPORT      - MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
 INCLUDE EDXALU           *30*  EDL INSTRUCTION EMULATOR
*-------------------------------------------------------------------
*  OPTIONAL FUNCTION SUPPORT - MUST BE IN PARTITION 1
*-------------------------------------------------------------------
 INCLUDE RLOADER          *17*  RELOCATING PROGRAM LOADER
*-------------------------------------------------------------------
*  DISK(ETTE) SUPPORT    - MUST BE GROUPED TOGETHER
                         - MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
*-------------------------------------------------------------------
 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 $EDXNUCT,EDX002 REPLACE END
*-------------------------------------------------------------
*
```

Figure 5-7. LINKCNTL Data Set for the Sample System

4(h) Enter MENU on the COMMAND INPUT line to return to the primary option menu. Then select option **8** and press the enter key to end $FSEDIT.

Step 5 - Edit DMEXTBL2 (Optional)

If you included dynamic data set extent support and you need to change the size of the DMEXTBL data set, you must edit the source module DMEXTBL2. In "Step 4 - Estimate the Size of the Dynamic Data Set Extents Table (DMEXTBL)" on page 4-33, you should have calculated the size of the table data area and entered the result in the space provided below. If not, you need to do that now.

REQUIRED SIZE FOR THE PRIMARY DATA AREA = _____BYTES

REQUIRED SIZE FOR THE EXTENT DATA AREA = _____BYTES

To edit DMEXTBL2:

5(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    nnP, LP=nnnn, PART=n
```

Once loaded, $FSEDIT displays the primary option menu.

5(b) Read a copy of the DMEXTBL2 data set into the work data set.

1. Enter a **3** on the OPTION line.
2. Enter **DMEXTBL2** on the DATA SET NAME line.
3. Enter **EDX002** on the VOLUME NAME line.

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

DATA SET NAME ==========> DMEXTBL2            (CURRENTLY IN WORK FILE)
VOLUME NAME ===========> EDX002
```

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

```
nn   LINES READ FROM   DMEXTBL2,EDX002
```

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

5(c) Enter a **2** (EDIT) on the OPTION line to edit DMEXTBL2. the OPTION line.

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

DATA SET NAME ==========> DMEXTBL2            (CURRENTLY IN WORK FILE)
VOLUME NAME ===========> EDX002
```

Press the enter key. $FSEDIT displays the contents of DMEXTBL2. The following example illustrates the data area you can change:

```
        COPY DMEXTBL1  -  TABLE HEADER INFO - 32 BYTES
PRMSTRT EQU  *       PRIMARY AREA OF TABLE - 920 BYTES
        DC   128F'0'         * USER MAY MODIFY
        DC   128F'0'         * USER MAY MODIFY
        DC   128F'0'         * USER MAY MODIFY
        DC    76F'0'         * USER MAY MODIFY
EXTSTRT EQU  *       EXTENT AREA OF TABLE - 2000 BYTES
        DC   128F'0'         * USER MAY MODIFY
        DC   128F'0'         * USER MAY MODIFY
        DC   128F'0'         * USER MAY MODIFY
        DC   128F'0'         * USER MAY MODIFY
        DC   128F'0'         * USER MAY MODIFY
        DC   128F'0'         * USER MAY MODIFY
        DC   128F'0'         * USER MAY MODIFY
        DC   104F'0'         * USER MAY MODIFY
        COPY DMEXTBL3
```

Figure 5-8. DMEXTBL2 Data Set Example

5(d) Edit DMEXTBL2.

To change the size of the data area, add or subtract the appropriate number of words in the section you are changing. To make the table larger, add another DC nnF'0' where *nn* is the number of words required. To make the table smaller, delete one of the DC statements or change the size of one of the DC statements.

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

```
EDIT --- DMEXTBL2,EDX002    336( 543)-----------------COLUMNS 001 072
COMMAND INPUT ===> MENU                               SCROLL ===>HALF
          .
          .
          .
```

5(f) Save the changed data set.

1. Enter a **4** on the OPTION line.
2. Enter the data set name and volume 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 =
                                                  PRESS PF3 TO EXIT
OPTION ===> 4

DATA SET NAME ==========> DMEXTBL2              (CURRENTLY IN WORK FILE)
VOLUME NAME ===========> EDX002
                .
                .
                .
WRITE TO DMXTBL2 ON EDX002 (Y/N)?  Y
```

$FSEDIT saves the contents of the work data set in DMEXTBL2 and issues the following message:

```
nn LINES WRITTEN TO DMEXTBL2,EDX002
```

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

5(g) Enter MENU on the COMMAND INPUT line to return to the primary option menu. Then select option **8** and press the enter key to end $FSEDIT.

5(h) Load $EDXASM and compile DMEXTBL2 as DMEXTBL.

1. Enter **DMEXTBL2** on the SOURCE (NAME,VOLUME) line and press the enter key. VOLUME must be the volume where all the other supervisor object modules reside (XS6006 in this example).
2. Enter the name of your work data set (in this example, ASMWORK) and press the enter key.
3. Enter **DMEXTBL** on the OBJECT (NAME,VOLUME) line and press the enter key.
4. Press the enter key in response to the SELECT OPTIONS prompt. Your compiled listing will print on $SYSPRTR. (Refer to the *Operator Commands and Utilities Reference* for other $EDXASM options.)

```
> $L $EDXASM,ASMLIB
SOURCE  (NAME,VOLUME): DMEXTBL2
WORKFILE(NAME,VOLUME): ASMWORK
OBJECT  (NAME,VOLUME): DMEXTBL,XS6006
LOADING $EDXASM    nnP, HH.MM.SS, LP=nnnn, PART=x
SELECT OPTIONS (?):
```

When you link the supervisor, $XPSLINK will include the modified extent table.

Step 6 - Edit SCBTBL - Segmentation Register Control Block Table (Extended Address Mode Only)

If you included extended address mode support and you need to change the size of the segmentation register control block table, you must edit the source module SCBTBL. You can increase or decrease the number of segmentation register control blocks (SCBs) in the data set. The default size is 113 SCBs. Use the $STGUT1 utility to monitor the use of SCBs.

To edit SCBTBL:

6(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    nnP, LP=nnnn, PART=x
```

Once loaded, $FSEDIT displays the primary option menu.

6(b) Read a copy of the SCBTBL data set into the work data set.

1. Enter a **3** on the OPTION line.
2. Enter **SCBTBL** on the DATA SET NAME line.
3. Enter **ASMLIB** on the VOLUME NAME line.

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

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

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

```
nn   LINES READ FROM   SCBTBL,EDX002
```

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

6(c) Enter a **2** (EDIT) on the OPTION line to edit SCBTBL.

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

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

Press the enter key. $FSEDIT displays the contents of SCBTBL. The following example illustrates the data area you can change:

```
         ENTRY   SCB0            ADDRESS OF FIRST SCB
*
         DC   A(TBLSIZE)      SIZE OF TABLE IN BYTES
SCB0     EQU  *
         DC   128F'0'         * USER MAY MODIFY
         DC   128F'0'         * USER MAY MODIFY
         DC   128F'0'         * USER MAY MODIFY
         DC   128F'0'         * USER MAY MODIFY
         DC   128F'0'         * USER MAY MODIFY
         DC   128F'0'         * USER MAY MODIFY
         DC   128F'0'         * USER MAY MODIFY
         DC   128F'0'         * USER MAY MODIFY
TBLSIZE  EQU  *-SCBTBL        TABLE SIZE IN BYTES
         $ID
         END
```

Figure 5-9. SCBTBL Data Set Example

6(d) Edit SCBTBL.

To change the size of the data area, add or subtract the appropriate number of words for the data area that starts at SCB0 and ends at TBLSIZE. To make the table larger, add another DC nnF'0' where *nn* is the number of words required. To make the table smaller, delete one of the DC statements or change the size of one of the DC statements.

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

```
EDIT --- SCBTBL,EDX002    336( 543)--------------------COLUMNS 001 072
COMMAND INPUT ===> MENU                                SCROLL ===>HALF
          •
          •
          •
```

6(f) Save the changed data set.

1. Enter a **4** on the OPTION line.
2. Enter the data set name and volume 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 =
                                                  PRESS PF3 TO EXIT
OPTION ===> 4

DATA SET NAME ==========> SCBTBL              (CURRENTLY IN WORK FILE)
VOLUME NAME ===========> EDX002
          •
          •
          •
WRITE TO SCBTBL ON EDX002 (Y/N)? Y
```

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

```
    nn LINES WRITTEN TO SCBTBL,EDX002
```

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

6(g) Enter MENU on the COMMAND INPUT line to return to the primary option menu. Then select option **8** and press the enter key to end $FSEDIT.

6(h) Load $EDXASM and compile SCBTBL.

1. Enter **SCBTBL** on the SOURCE (NAME,VOLUME) line and press the enter key.
2. Enter the name of your work data set (in this example, ASMWORK) and press the enter key.
3. Enter **SCBTBL** on the OBJECT (NAME,VOLUME) line and press the enter key. VOLUME must be the volume where all the other supervisor object modules reside (XS6006 in this example).
4. Press the enter key in response to the SELECT OPTIONS prompt. Your compiled listing will print on $SYSPRTR. (Refer to the *Operator Commands and Utilities Reference* for other $EDXASM options.)

```
> $L $EDXASM,ASMLIB
SOURCE  (NAME,VOLUME): SCBTBL
WORKFILE(NAME,VOLUME): ASMWORK
OBJECT  (NAME,VOLUME): SCBTBL,XS6006
LOADING $EDXASM    nnP, HH.MM.SS, LP=nnnn, PART=n
SELECT OPTIONS (?):
```

When you link the supervisor, $XPSLINK will include the modified table.

Step 7 - Edit $JOBUTIL Procedure File

You are ready now 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.

To perform the assembly and link edit in one step, you can use the $JOBUTIL job stream processing utility with a set of job-control statements that IBM supplies. The job-control statements instruct $JOBUTIL to load and execute the $EDXASM compiler and the $XPSLINK linkage editor. These statements also provide the necessary data set names for the operation. The job-control statements reside in the $SUPPREP data set 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 7(b) through 7(d). 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 7(a) and 7(e).

Once you copy $SUPPREP to SUPPREPS on EDX002, proceed to step 8. 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 the example on the following page that shows the $SUPPREP data set.

To edit $SUPPREP:

7(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    nnP, LP=nnnn, PART=x
```

Once loaded, $FSEDIT displays the primary option menu.

7(b) Read a copy of 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.

If you do not need to edit $SUPPREP, skip to step 7(d).

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

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

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

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

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

7(c) Enter a **2** (EDIT) on the OPTION line 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.

```
$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 6, MODIFICATION LEVEL 0
00040 *
00050 LOG      $SYSPRTR
00060 JOB      $SUPPREP
00070 REMARK   ** ENTER -GO- AFTER -XS6006- 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 6 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
```

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

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

Some of these names appear on more than one line.

7(e) 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 6, MODIFICATION LEVEL 0
```

$FSEDIT returns to the primary option menu.

7(f) Save $SUPPREP as SUPPREPS.

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.
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,

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

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

7(g) Enter MENU on the COMMAND INPUT line to return to the primary option menu. Then select option **8** and press the enter key to end $FSEDIT.

```
$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 8 - Execute SUPPREPS

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 and convert the results to an executable program using $XPSLINK. To perform these operations, execute SUPPREPS as follows:

Note: You can use the session manager option 2.8 or 2.13, but you must have installed the program preparation modules (5719-XX7). For general information on how to use the session manager, refer to the *Operation Guide*.

8(a) Press the attention key and load the $JOBUTIL utility. Respond to the prompts as shown.

```
> $L $JOBUTIL
DS1 (NAME,VOLUME): SUPPREPS,EDX002
LOADING $JOBUTIL      nP, HH.MM.SS, LP=nnnn, PART=x
REMARK   ** ENTER -GO- AFTER -XS6006- HAS BEEN VARIED ONLINE **
REMARK   ** PRODUCES $EDXNUCT ON EDX002 **

PAUSE-*-ATTN:GO/ENTER/ABORT

PAUSE
```

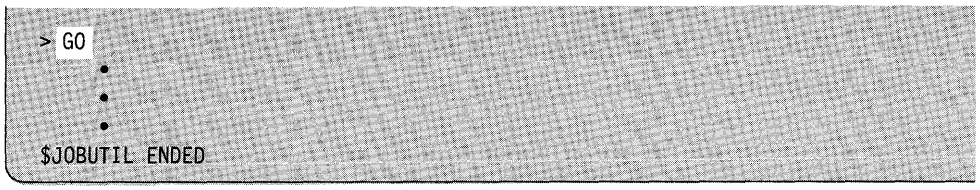
8(b) Insert diskette XS6006 into your diskette unit. Diskette XS6006 contains the supervisor object modules that are accessed only during the system generation process.

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
XS6006 ONLINE ON SLOT 1
```

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



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

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 link-control 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 link-control data set produced as a result of processing $XPSPRE prints on $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 the link-control data set, 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.

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 $EDXNUCT, a supervisor data set allocated automatically by $XPSLINK. $XPSLINK and $JOBUTIL end.

When $JOBUTIL ends, 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 whatever you named it), 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."
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 supervisor module names and entry points is included in Appendix D, "Supervisor Module Names (CSECTS)" on page D-1. If you have not included the module listed, edit the link-control data set to include the missing module.
5. If your supervisor exceeded 64K within a specific partition, 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 4-33.

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 9 - Edit $SRPROF - (Extended Address Mode Only)

You can edit the $SRPROF configuration profile data set at any time to change the mixture of static and dynamic partitions. At IPL time, the system reads in $SRPROF and sets up the tables required to allocate, manage, and release the dynamic I/O segmentation registers.

Note: If you do not need to edit $SRPROF, skip to "Step 10 - Initialize Your Tailored Operating System" on page 5-46.

9(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    nnP, LP=nnnn, PART=x
```

9(b) Read a copy of the $SRPROF data set into the work data set.

1. Enter a **3** on the OPTION line.
2. Enter **$SRPROF** 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 ==========> $SRPROF          (CURRENTLY IN WORK FILE)
VOLUME NAME ===========> ASMLIB
```

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

```
nn   LINES READ FROM   $SRPROF,ASMLIB
```

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

9(c) Enter a *2* (EDIT) on the OPTION line to edit $SRPROF.

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

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

Press the enter key. $FSEDIT displays the $SRPROF data set.

If errors occur when the system tries to read this data set or if you fail to edit the data set, the system uses the defaults listed in Figure 5-10 on page 5-42. An explanation of the operands follows the figure.

```
*                    EVENT DRIVEN EXECUTIVE                          *
*                  EXTENDED ADDRESS MODE SUPPORT                     *
*                 $SRPROF - IPL CONFIGURATION FILE                   *
*                 VERSION 6 - MODIFICATION LEVEL 0                   *
*                                                                    *
* THE FOLLOWING DEFINES THE DEFAULT CONFIGURATION FILE FOR EDX       *
* EXTENDED ADDRESS MODE SUPPORT.  FOR A COMPLETE DESCRIPTION OF      *
* THESE STATEMENTS REFER TO THE 'IBM SERIES/1 EVENT DRIVEN EXECUTIVE *
* INSTALLATION AND SYSTEM GENERATION GUIDE,' SC34-0936.              *
*                                                                    *
* STATEMENTS WITH AN ASTERISK (*) IN COLUMN 1 ARE IGNORED.           *
*                                                                    *
**********************************************************************
* MAY BE 'S' OR 'D'.  BEARS A ONE-TO-ONE CORRELATION TO THE PARTS    *
* OPERAND ON THE SYSTEM STATEMENT FOR PARTITIONS 1 TO 8.  INDICATES  *
* WHETHER THE PARTITION IS STATIC 'S' OR DYNAMIC 'D'.                *
*  - PARTITION 1 MUST BE STATIC.                                     *
*  - PARTITIONS 9 THROUGH 32 ARE ALWAYS DYNAMIC.                     *
**********************************************************************
*
*      12345678
 PARTS=SDDDDDDD
*
**********************************************************************
* SPECIFY TWO SUBITEMS RELATIVE TO THE EDX LOADER:                   *
* 1) 'S' OR 'A' FOR PART=ANY. ON THE LOAD INSTRUCTION, WILL ATTEMPT TO *
*             LOAD IN JUST 'STATIC' PARTITIONS OR IN 'ANY' OF THE    *
*             PARTITIONS (STATIC OR DYNAMIC).                        *
* 2) 'F' OR 'R' FOR PART=ANY, STATIC, OR DYNAMIC.  ON THE LOAD       *
*             INSTRUCTION, WILL ATTEMPT TO LOAD GOING 'FORWARD'      *
*             (PARTITIONS 1-32) OR IN 'REVERSE' (PARTITIONS 32-1).   *
**********************************************************************
*
 LOADER=(S,F)
*
**********************************************************************
* MAY BE 'Y' OR 'N'.  SPECIFIES WHETHER THE I/O SEG REG MANAGER      *
* ROUTINE "SRMGR" WILL WAIT FOR I/O SEG REG ALLOCATION OR TERMINATE IF *
* THERE ARE NO SCB OR NOT ENOUGH CONTINUOUS SEG REGS AVAILABLE FOR AN *
* AN I/O REQUEST.                                                    *
*   (Y) WAIT FOR I/O SEG REG ALLOCATION                              *
*   (N) DON'T WAIT, TAKE THE TASK ERROR EXIT IF CODED, OR THE TASK WILL*
*       BE TERMINATED IF NOT CODED.                                  *
**********************************************************************
*
 WAITIOSR=Y
*
```

Figure 5-10. Default $SRPROF Data Set

Operand	*Description*
PARTS=	Can be an S (static) or D (dynamic). Partition 1 must be static. Partitions 2 through 8 can be static or dynamic. Partitions 9 through 32 can be dynamic only. (See step 9(d) for an explanation of static and dynamic.) The default is PARTS=SDDDDDDD.
LOADER=	Instructs the EDX loader how to load programs through two subitems. The first subitem can be S (static) or A (any). The EDL LOAD instruction with PART=ANY operates according to what you code for this subitem. If you specify S, the loader attempts to load programs into static partitions only. If you specify A, the EDX loader attempts to load programs into any partition, static or dynamic. The second subitem can be F (forward) or R (reverse). If you specify F, the loader attempts to load programs starting with partition 1. If you specify R, the loader attempts to load programs starting with the partition specified on the NUMPART= operand of the SYSPARTS statement. The default is LOADER=(S,F).
WAITIOSR=	Can be Y (yes) or N (no). If you specify Y, the I/O segmentation register manager routine (SRMGR) waits for I/O segmentation register allocation. This option will cause long waits if I/O segmentation registers or any I/O resources are unavailable. If you specify N, the manager does not wait but ends the task and issues an error message to $SYSLOG. The default is WAITIOSR=Y. For an explanation of how to override what is coded in WAITIOSR, refer to $TCBFLGS in the *Customization Guide*.

9(d) Decide if you want to assign a partition as static or dynamic.

In static partitions, the system assigns I/O segmentation registers to all user and common areas at initialization time. In dynamic partitions, the system must allocate I/O segmentation registers for each I/O request you issue and deallocate them when the I/O operation completes. A static 64K-byte partition requires 32 I/O segmentation registers unless part of a supervisor occupies the partition. Partition 1 must be static and does not require 32 I/O segmentation registers. Even if you specify partition 1 as dynamic, the system maps it as static. The system maps it for I/O from address 0000 to the end of the system definition statements ($EDXDEF). The system also maps the user area at the end of partition 1 for I/O.

Partitions 9 through 32 must be dynamic, and partitions 2 through 8 can be either static or dynamic.

I/O operations to a static partition are faster than I/O operations to a dynamic partition since the system does not have to allocate and deallocate I/O segmentation registers. However, the more partitions that you assign as static, the fewer dynamic segmentation registers the system has available for I/O operations. If you assign too many partitions as static and programs that perform considerable I/O are executed in the dynamic areas, then those programs can fail to execute because there are no I/O segmentation registers available. They go into a wait state or the system ends the programs then issues an error message to $SYSLOG (depending on what you code for the WAITIOSR operand in the $SRPROF data set).

Note: Refer to the *Customization Guide* for information on changing the mapped area in each partition.

You can assign 8 static partitions, but then you would have only the supervisor area within the partitions available for dynamic allocation. Consequently, tasks performing I/O to dynamic partitions would be either "waiting" continually or the system would end them, depending on what you specify for the WAITIOSR operand. If a task tries to perform I/O to a dynamic partition and the size of the I/O buffer is larger than any block of segmentation registers that will ever be available, the system ends the program and issues an error message to $SYSLOG.

If some of your programs perform loads with PART=ANY, you can load these programs into static partitions only by taking the default of "S" for the first operand on the LOADER statement in $SRPROF. However, then your static partitions will fill up with programs that do not need to run in static partitions. Therefore, change the loads with PART=ANY to PART=STATIC (for programs you want to run in static partitions) or PART=DYNAMIC (for programs you want to run in dynamic partitions).

If you do ***not*** want to change the PART=ANY operand in the LOAD instruction, assign the static partitions that you need in sequential order starting with partition 2 and going up. You then force all the loads with PART=ANY to try to load in all the dynamic partitions first before trying the static partitions, leaving the static partitions available for the programs that you want to run in static partitions.

9(e) Edit $SRPROF using the following rules:

- Each line can contain only one operand.
- An operand must be the first entry on the line.
- An operand can start in any column.

You can place comments after the operand. You can also make an entire line a comment by placing an asterisk in column 1.

The following example specifies partitions 1 through 4 as static and partitions 5 through 8 as dynamic:

Example: $SRPROF data set.

```
PARTS=SSSSDDDD
LOADER=(A,R)
WAITIOSR=N
```

After you edit the $SRPROF data set, you need to save your changes.

9(f) Enter MENU on the COMMAND INPUT line and press the enter key to end EDIT mode as shown.

```
EDIT --- $SRPROF,ASMLIB    53 ( 543)------------------COLUMNS 001 072
COMMAND INPUT ===> MENU                                SCROLL ===>HALF
          •
          •
          •
```

9(g) Save the changed data set.

1. Enter a **4** on the OPTION line.
2. Enter name of the data set name and volume 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 ==========> $SRPROF              (CURRENTLY IN WORK FILE)
VOLUME NAME ===========> EDX002
          •
          •
          •
WRITE TO $SRPROF ON EDX002 (Y/N)?  Y
```

$FSEDIT issues the following message:

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

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

Step 10 - Initialize Your Tailored 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 $EDXNUCT on EDX002.

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

```
> $L $INITDSK
LOADING $INITDSK    nnP, LP=nnnn, PART=x

$INITDSK - DISK INITIALIZATION UTILITY

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

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

Now you can use $EDXNUCT as your supervisor.

10(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 performing an IPL from diskette XS6001. Then use the II command of $INITDSK to redirect the IPL text to point to $EDXNUC on EDX002. Repeat steps 2 through 5 to correct any errors.

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

```
*** EVENT DRIVEN EXECUTIVE *** V6.0
***          XPS           ***
IPL=$EDXNUCT,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    26880  38656    0      0   6900  6900   9700  10000
 2    23808  41728    0      0   5D00  5D00   A300  10000
 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
                    •
                    •
                    •
```

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 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.

Now you can 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 10(c) and 10(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 11 - Verify the System Generation Process (Optional)" on page 5-49.

Note: For information on maintaining more than one supervisor, see "Maintaining Multiple Supervisors" on page 5-51.

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

```
> $L $COPYUT1
LOADING $COPYUT1     nnP, LP=nnnn, PART=x

$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:  $EDXNUCT
ENTER TO (TARGET) MEMBER OR * FOR SAME NAME AS SOURCE:  $EDXNUC
COPY COMPLETE            257 RECORDS COPIED

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

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

```
> $L $INITDSK
LOADING $INITDSK    nnP, LP=nnnn, PART=x

$INITDSK - DISK INITIALIZATION UTILITY

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

IPL TEXT WRITTEN

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

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

Step 11 - Verify the System Generation Process (Optional)

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

11(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      nnP, LP=nnnn, PART=x


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 CALCSRC successfully (completion code = −1), continue. If you receive a completion code other than −1, refer to the *Messages and Codes* book for its meaning, correct the problem, and reassemble CALCSRC.

11(b) Press the attention key, load $EDXLINK and provide the parameters necessary to link edit CALCDEMO. 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     nnP, LP=nnnn, PART=x
$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 CALCDEMO.

11(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     nP, LP=nnnn, PART=n
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.

Use the LS command of the $IOTEST utility to list the I/O devices supported by the generated supervisor. With this list, you can verify that the supervisor actually supports 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. If your supervisor resides on a volume other than EDX002, the system searches that volume for the system utilities. Since the system utilities reside on volume EDX002, each time you accessed the system utilities, you would need to indicate that EDX002 is the volume where they reside.

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

```
COMMAND (?):  II
NUCLEUS:  $EDXNUCT
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.
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, $LOG, $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.

i. $LOG will allocate a 200-record data set (EDXLOGDS) if it has not been allocated already.

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 Executive (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     nnP, LP=nnnn, PART=x

$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 (?):
```

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

```
> $L $COPY
LOADING $COPY     nnP, LP=nnnn, PART=x

$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      nnP, LP=nnnn, PART=x

$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 (?):
```

Install Local Communications Controller Microcode Patches

When you receive the LCC hardware or an engineering change (EC) to the hardware, you might also receive microcode patches. If so, you will receive a diskette in basic exchange format that contains these patches to the microcode. You must copy the diskette to a data set named $LCCECnn, where *nn* is the EC level number.

Use the $DISKUT1 utility to allocate the data set on your IPL volume. Then use the $COPY utility to copy the diskette to the data set.

Example: Allocate a data set for EC level 02 and copy the microcode patch diskette.

```
> $L $DISKUT1
LOADING $DISKUT1      nnP, HH.MM.SS, LP=nnnn, PART=x
$DISKUT1 - DATA SET MANAGEMENT UTILITY I

USING VOLUME EDX002

COMMAND (?):  AL $LCCEC02 3 D
COMMAND (?):  EN

> $L $COPY
COMMAND (?):  RE
SOURCE ($$EDXVOL,VOLUME):  $$EDXVOL,RAM41
TARGET (NAME,VOLUME):  $LCCEC02,EDX002
SPECIFY START/END (Y/N)?  N
ENTER BASIC EXCHANGE DATA SET NAME:  RAMDATA1
NUMBER OF RECORDS COPIED:  3
COPY COMPLETED
COMMAND (?):  EN
```

Chapter 6. Migrate to Version 6.0

This chapter describes what you must do to convert from Version 1 or 2 to Version 6.0. Your programs and data must be converted before you can use them with Version 6.0.

Conversion of Programs

Before using EDX Version 6.0, you must recompile or reassemble all EDX Version 1 or 2 programs. The source code must be modified if the Version 6.0 function is to be utilized or if the program references system control blocks.

In Version 6.0, 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 6.0 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 terminals and will now be used with 3101 or equivalent terminals, then you should read the Version 6.0 *Language Reference* if you wish to make coding changes to the program.

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 6.0; 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. Words 3 and 4 of the request block are a doubleword in Version 6.0. 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 for disk and diskette in Version 6.0 differs from 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 6.0 format before they can 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 6.0 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 6.0, an updated copy of $MIGAID must be added to the EDX system. The updated copy of $MIGAID can be obtained through your local IBM Representative. 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 6.0 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 6.0 system, reads the diskettes of the ***save set*** and produces, on disk, the Version 6.0 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 6.0 unless they are recompiled.

$MIGCOPY, executing on a Version 6.0 system, converts data sets on Version 1 or 2 diskettes to Version 6.0 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 6.0 data management environment:

1. Create an diskette containing the Version 1 or 2 $EDXNUC and starter utilities from which you can IPL.
2. Create a save set from the Version 1 or 2 disk (4962 or 4963) environment with $MIGAID.
3. After installing EDX Version 6.0, 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 can 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 6-6.

The $MIGAID save set and the diskettes from which you can IPL, are also the means for returning to the original environment, should that need arise.

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 Representative.
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.

Use the following procedure to ensure that you can return to Version 1 or 2 as well as proceed to Version 6.0:

1. Make a Version 1 or 2 diskette that contains the minimum starter utilities from which you can IPL. 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 copy individually 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
```

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 source 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 6.0 Program Directories.
9. Run $MIGRATE to copy the diskettes to your Version 6.0 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 can remove the IPL diskette at this time; insert the saved diskettes as prompted by $MIGAID.

Disk Conversion Procedure Using Diskettes Produced by $COPYUT1 or $COPY

Use the following procedure to ensure that you can return to Version 1 or 2 as well as proceed to Version 6.0:

1. Make a Version 1 or 2 diskette that contains the minimum starter utilities from which you can IPL. If you have two diskette drives, you can 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 source 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 6.0 Program Directory.
10. Run $MIGCOPY to copy your Version 1 or 2 diskettes to your Version 6.0 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.
    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 6.0:
   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 6.0 system.
   b. After you have installed Version 6.0, use the Version 6.0 $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 6.0, 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 following).
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 can 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 6.0.

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 restore your current environment or place selected contents of the diskettes on your Version 6.0 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      nnP, HH.MM.SS, LP=nnnn, PART=x
$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 (Y/N)?  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 D
MIGWORK CREATED

COMMAND (?):  EN

$DISKUT1 ENDED AT 13:15:57
```

To clear the data set:

```
> $L $DISKUT2
LOADING $DISKUT2      nnP, HH.MM.SS, LP=nnnn, PART=x
$DISKUT2  - DATA SET MANAGEMENT UTILITY II

USING VOLUME EDX002

COMMAND (?):  CD MIGWORK
CLEAR ENTIRE DATA SET (Y/N)?  Y

ARE ALL PARAMETERS CORRECT (Y/N)?  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
LOADING $MIGAID     nnP, HH.MM.SS, LP=nnnn, PART=x
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.

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 end $MIGAID. You can enter EN 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 ended 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.

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
LOADING $SSINIT     nnP, HH.MM.SS,  LP=nnnn,  PART=x

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 (Y/N)? 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 (Y/N)?
```

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 required to do the save described 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
  1  DLETEMP1(PART)

MIGACDS - DATA SETS COPIED FROM LIB002 TO X02
  2  DLETEMP1  1805    DLETEMP    1803

MIGACDS - DATA SETS COPIED FROM LIB002 TO X03
  3  DLETEMP2(PART)

MIGACDS - DATA SETS COPIED FROM LIB002 TO X04
  4  DLETEMP2  1806    DLETEMP3    1802

MIGACDS - DATA SETS COPIED FROM LIB002 TO X05
  5  DMINTRFS 273  DMINTRFO 60 $MIGAID 64 $MIGAIDO 100

  5  $MIGAIDL 150  $MIGAIDC  2 TSTUT3  42 $MIGAIDS 679


MIGADSC - OPERATION SUCCESSFULLY COMPLETED.
```

1 Indicates the first part of DLETEMP1 is on diskette X01.

2 Indicates the remainder of DLETEMP1 and DLETEMP are on diskette X02.

3 Indicates the first part of DLETEMP2 is on diskette X03.

4 Indicates the remainder of DLETEMP2 and DLETEMP3 is on diskette X04.

5 Shows you 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     nnP, HH.MM.SS, LP=nnnn,  PART=x

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 (?):
```

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.

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       nnP, HH.MM.SS, LP=nnnn, PART=x
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.
```

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 6.0, converts $MIGAID output to Version 6.0 format. If you are not converting to Version 6.0 from a previous version of EDX, you do not need this program.

Note: $MIGRATE (on EDX Version 6.0) 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 Representative.
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 load $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 6.0 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 6.0 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 6.0.
- It does not copy program-type data sets (Version 1 or 2 programs will not run on Version 6.0 unless they are recompiled).
- It has an "LA" command similar to $DISKUT1 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 6.0 target volume.

CALL Copy All. Copies all data members from the Version 1 or 2 source volume to the Version 6.0 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 6.0 format. For example:

```
> $L $MIGCOPY
$MIGCOPY        nnP, HH.MM.SS, LP=nnnn, PART=x
SOURCE VOLUME:  RECOVR
TARGET VOLUME:  EDX005
```

The source and target volumes can be changed later by using the CV command.

Example 2 - Using the List All (LA) Command: The LA command can 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 6.0 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, copy 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
```

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 that will prompt you 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 communications line
- DISK – Defines direct access storage devices
- EXIODEV – Defines EXIO interface devices
- HOSTCOMM – Defines host communication support
- LCC – Defines a Local Communications Controller attachment
- SENSORIO – Defines sensor I/O devices
- SYSCOMM – Defines base and common storage partitions
- SYSEND – Defines the end of the system partition statements
- SYSMSG – Defines system message destination
- SYSPARMS – Defines the sizes of system buffers and the partition in which to load programs
- SYSPARTS – Defines the number of partitions in the system
- TAPE – Defines tape device
- TERMINAL – Defines terminals
- TIMER – Defines system timer feature.

The system generation process uses these statements as described in Chapter 5, "Generate a Tailored Operating System."

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 terminals 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.

Syntax:

label	**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 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. **Note:** If your list contains only one device, you do not need to enclose the list in parentheses.
END =	YES, for the last ADAPTER statement in the system definition module. The default is END = NO.

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
```

Example 3: A Multidrop Work Station Attachment (SMIO) at address 80 with two devices attached.

- One 4980 terminal at address 80
- One 4980 terminal 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 Communications Line

BSCLINE defines a binary synchronous communications 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 X.21 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.

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, refer to 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, refer to 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.

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.

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 communications 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 communications line at address X'58' attached to a Series/1 through an MFA. Specifies a list of three poll/select addresses.

```
BSC1    BSCLINE ADDRESS=58,ADAPTER=MFA,TYPE=MT,POLL=(C0,C1,C2)
```

Example 3: A binary synchronous communications line at address X'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 communications line at address X'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
IDSK	Disk and Diskette Unit
4964	4964 Diskette Unit
4965	4965 Diskette Unit within the 4952 Model C or 30D processor, the 4954 or 4956 Model 30D, 60D, or 60E 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)
4963-64	4963-64 Disk (64-megabyte unit)
4963-58	4963-58 Disk (58-megabyte unit with fixed heads)
4967-2	4967 model 2CA 200-megabyte unit (one or two 200 megabyte units on the same attachment card)
4967-2A	4967 model 3CA 358-megabyte unit (one or two 358 megabyte units on the same attachment card)

4967-4 4967 model 2CB 200-megabyte unit (three or four 200 megabyte units on the same attachment card)

4967-4A 4967 model 3CB 358-megabyte unit (three or four 358 megabyte units on the same attachment card)

DDSK-30 30-megabyte disk.

DDSK-60 60-megabyte disk.

ADDRESS= The hexadecimal address of the unit.

Notes:

1. If you have an IDSK disk or diskette unit, the base device address must end in either 0 or 8 (for example, X'50' or X'38'). The first disk must be at the base address of the attachment. The first diskette must be at the base address + 1. For example, if the base address is X'60', the first disk must be at address X'60' and the first diskette must be at address X'61'.
2. If you have a 4965 diskette unit within the 4952 Model 30D processor, the 4954 or 4956 Model 30D/60D/60E 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'.
3. 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.
4. If the device is to be used for a stand-alone dump, the device address must be between X'00' and X'45'. Address X'61' is valid for IDSK only.

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 4-11 for an explanation of performance volumes.

Each volume you specify requires 46 bytes of storage.

Notes:

1. You cannot designate performance volumes on a diskette. As a result, this operand is ignored if DEVICE=4964, 4965, or 4966.
2. This operand is optional; you must allocate the volumes with the $INITDSK utility.
3. Volume names, tape IDs, and TERMINAL statement labels must be unique. No tape ID or TERMINAL statement label can match a volume name.

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 statements defining the 60-megabyte disk and an optional 4965 diskette unit.

```
DISK  DEVICE=DDSK-60,ADDRESS=44
DISK  DEVICE=4965,ADDRESS=45,END=YES
```

Example 4: DISK definition statements defining an IDSK disk.

```
DISK  DEVICE=IDSK,ADDRESS=60,VOLNAME=(PRFRM1,PRFRM2),     C
      TASK=YES
DISK  DEVICE=IDSK,ADDRESS=61,END=YES
```

Example 5: DISK definition statements defining the model 4967-2 and 4967-4A disks.

```
DISK  DEVICE=4967-2,ADDRESS=48
DISK  DEVICE=4967-4A,ADDRESS=60,VOLNAME=(MNOPQR,STUVWX),  C
      TASK=YES
DISK  DEVICE=4967-4A,ADDRESS=61
DISK  DEVICE=4967-4A,ADDRESS=62,END=YES
```

EXIODEV - Define EXIO Interface Device

EXIODEV defines the devices to be supported through 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 (5719-SX1) Version 1.
- Physical unit that defines an SDLC line for the Series/1 Systems Network Architecture (5719-XX9) Version 2.
- Physical unit that defines an SDLC line for the Series/1 Primary Systems Network Architecture (5719-XT4).
- Physical unit that defines an SDLC line for Advanced Program-to-Program Communications (APPC).

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) Version 1, or Series/1 Systems Network Architecture (5719-XX9) Version 2, or the Series/1 Primary Systems Network Architecture (5719-XT4).

For more information on defining SDLC device configuration records for APPC, Primary Systems Network Architecture (PSNA), or Systems Network Architecture (SNA) using shared SDLC, refer to the Network Definition Utility Guide (5719-XT5).

Notes:

1. Any device defined through 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 results.

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 device being used by SNA support, this address must match the address specified for the ADDRESS= operand of the SNAPU definition statement. For a device being used by APPC or PSNA, this address must match the address specified in the SDLC device configuration record.
MAXDCB=	The maximum number of chained DCBs that will be used for this device. The value specified must be sixteen 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. For a device being used by the Series/1 Systems Network Architecture (5719-SX1) Version 1, or Series/1 Systems Network Architecture (5719-XX9) Version 2 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 operand of the network control program (NCP). For a device being used by the Series/1 APPC support, the Series/1 SNA using shared SDLC support, or the Series/1 PSNA support, specify the value 16. 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 device being used by EDX APPC, SNA, or PSNA 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=14,RSB=4,MAXDCB=16,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
```

LCC - Define a Local Communication Controller Attachment

The LCC statement defines a Local Communications Controller attachment to the EDX supervisor. The LCC statement must contain the base (ring) address of the attachment. That address must be a multiple of four. With the LCC statement, you define all three subchannels of the device:

- Subchannel 0 – base address
- Subchannel 1 – base address + 1
- Subchannel 2 – base address + 2.

The system requires three LCC statements for each attachment. Group all LCC statements together in the $EDXDEF data set. The last LCC statement must specify END=YES. (Refer to the *Communications Guide* for LCC programming examples.)

Syntax:

label	**LCC**	**ADDRESS=,END=**
Required:	**ADDRESS=**	
Defaults:	**END=NO**	

Operand	*Description*
label	A 1 to 8 alphanumeric character label beginning with a letter or one of the following special characters: $, #, or @.
ADDRESS=	The hexadecimal address of the Local Communications Controller Attachment subchannel.
END=	YES, for the last LCC statement in the system definition module. The default is END=NO.

Example: Two Local Communications Controller attachments, one starting at address X'50' and one starting at address X'58'.

```
        LCC  ADDRESS=50
        LCC  ADDRESS=51
        LCC  ADDRESS=52
        LCC  ADDRESS=58
        LCC  ADDRESS=59
        LCC  ADDRESS=5A,END=YES
```

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.

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
```

SYSMSG - Define System Message Destination

By using the SYSMSG statement, you can direct messages that normally go to the $SYSLOG terminal to a disk data set or to the Communications Facility log. At IPL time the system will check if DISK=YES or CF=YES was coded on the SYSMSG statement. If it was coded, $VIRLOG will be loaded and will direct the messages to the desired destinations.

Syntax:

blank	**SYSMSG**	**TERM=,DISK=,CF=**
Required:	**None**	
Default:	**TERM=YES,DISK=NO,CF=NO**	

Operand	*Description*
TERM=	NO, if you do not want $VIRLOG to direct messages to the $SYSLOG terminal. YES is the default.
DISK=	YES, if you want $VIRLOG to direct messages to the disk data set, EDXSMLDS. NO is the default.
CF=	YES, if you want $VIRLOG to direct messages to the Communications Facility log. NO is the default. Specify DISK=YES or CF=YES if you want $VIRLOG loaded when you IPL.

Example: Specify that $SYSLOG messages be sent to the disk data set (EDXSMLDS), the $SYSLOG terminal, and the Communications Facility log.

```
SYSMSG   TERM=YES,DISK=YES,CF=YES
```

Notes:

1. If you specify CF=YES and/or DISK=YES along with TERM=YES, the system message will not be sent to the terminal if it is busy. It will be sent to the Communications Facility log and/or to disk.
2. Do not issue I/O to the $SYSLOG terminal that requires user response.
3. If you issue I/O to the $SYSLOG terminal, the IOCB attributes are ignored.
4. If you issue I/O directly to #SYSLOG to bypass $VIRLOG, any messages issued to $SYSLOG during the I/O operation will not go to the terminal.

SYSPARTS - Define Partitions

The SYSPARTS system partition statement defines the number of partitions, the number of programs allowed per partition, and the size of each partition for the generated system. The SYSPARTS statement must be specified once and must be the first statement in $EDXDEF.

Notes:

1. A maximum of 254 characters are permitted in the operand field. Therefore, if all the operands in the SYSPARTS statement are coded, the 254 maximum characters might be exceeded and an error could occur. Avoid this error condition by allowing operands to default whenever possible.
2. 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 SYSPARTS statement, the partition is defined automatically. However, this partition will contain supervisor code only and cannot be used for execution of application programs.

The EDX supports two types of storage: mapped and unmapped. Mapped storage is the storage you define with the PARTS= operand. Unmapped storage is any storage not defined with the PARTS= operand, or any storage above 2048K on the 4956 model J and K, above 1024K on the 4956 model E, 60E, or H, and above 512K for all other processors. The system acquires unmapped storage through the unmapped storage instructions. Refer to the *Language Reference* for a description of these instructions. For an explanation of using unmapped storage for application programs, refer to the *Event Driven Executive Language Programming Guide*.

Syntax:

blank	**SYSPARTS NUMPART=,PARTS=,MAXPROG=**
Required:	**NUMPART=**
Defaults:	**PARTS=32,MAXPROG=10**

Operand *Description*

NUMPART=

The number of partitions to create for the system. The range for NUMPART is 1 to the maximum number of partitions supported by your processor.

PARTS= The number of 2K (1K = 1024 bytes) blocks of storage to be assigned to each partition for the execution of application programs.

PARTS=nn

Where nn is the number of 2K blocks to be assigned to all of the partitions defined by the NUMPART= operand. The value specified for the PARTS= operand in this format must be greater than zero. The range for PARTS= is 0 to the maximum number of partitions supported by your processor.

PARTS=(nn,...,nn,nn,nn,nn,nn,nn,nn,nn,nn,nn,nn,nn,nn,nn)

Where nn is the number of 2K blocks to be assigned to each of the partitions. If memory is selected for a partition, the maximum number of programs for the corresponding partition must be greater than zero. This operand is selected with the NUMPART= operand. The number of entries specified on the PARTS= operand must be equal to the number of entries specified for the MAXPROG= operand, if the operands are in the same format.

You can specify a maximum of thirty-two 2K blocks of storage (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 required for supervisor code within a partition.

You can define up to:

- Thirty-two 64K partitions for the 4956 models J and K
- Sixteen 64K partitions for the 4956 models E, 60E, and H (with extended address mode defined)
- Eight 64K partitions for the 4955 and 4956 models B and G
- Four 64K partitions for the 4954
- 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 – 16 or 1 – 32 for extended address mode, 1 – 8 otherwise) must fit in address spaces or map areas that are numbered 0 – 15, 0 – 31, or 0 – 7, respectively, and can contain 64K of designated storage.

The amount of storage available in any partition is the smaller of:

- The size you define for the PARTS= operand
- 64K minus the size of the supervisor.

(See Chapter 4, "Select Your Required Support" 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 thirty-two 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 thirty-two 2K blocks is unmapped storage.

(To determine the storage required by licensed programs such as the Indexed Access Method and the Multiple Terminal Manager, see work sheet 4 in Appendix E.)

For examples of defining the SYSPARTS statement, see "Examples for the System Partition Statements" on page A-31.

MAXPROG=

The maximum number of concurrently executing programs allowed in a partition.

MAXPROG=nnn

Where nnn is the maximum number of programs in all of the partitions defined by the NUMPART operand. The value specified for the MAXPROG operand in this format must be greater than zero.

MAXPROG=

(nnn,...,nnn,nnn,nnn,nnn,nnn,nnn,nnn,nnn,nnn). Where nnn is the number of programs allowed for each partition. The number of entries specified on the MAXPROG= operand must be less than or equal to the number specified on the NUMPART= operand. The number of entries specified on the MAXPROG= operand must be equal to the number of entries specified for the PARTS= operand, if the operands are in the same format.

The range for nnn is 0 to 100. You must specify 0 for a partition that does not contain any storage. If a partition contains memory, the MAXPROG entry must be from 1 to 100.

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. 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.

SYSPARMS - Define Buffers and Partition

The SYSPARMS system partition statement defines the sizes of various system buffers and the partition in which to load the $LOG and $VIRLOG programs when you perform an initial program load (IPL) of the operating system. The SYSPARMS statement is optional. If the SYSPARMS statement is coded, it must follow the SYSPARTS statement and precede the SYSCOMM and SYSEND statements.

Note: A maximum of 254 characters are permitted in the operand field. Therefore, if all the operands in the SYSPARMS statement are coded, the 254 maximum characters might be exceeded and an error could occur. Avoid this error condition by allowing operands to default whenever possible.

Syntax:

blank	**SYSPARMS**	**DATEFMT=,IABUF=,INITMOD=, INITPRT=,LOGPART=,MECBLST=, TBPART=,VIRPART=,XPSSTK=**
Required:	**NONE**	
Defaults:	**DATEFMT=MMDDYY,IABUF=20,INITPRT=0,LOGPART=0, MECBLST=0,TBPART=0,VIRPART=0,XPSSTK=20**	

Operand *Description*

DATEFMT=

The format to be used when the date is displayed (PRINDATE or $W) or when entering the date using $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 set the date.

IABUF= The maximum number (in decimal) of interrupts that can 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. (Otherwise the system will stop or enter a continuous run loop.) Each increment increases the supervisor storage requirements by eight bytes. For example, the default value will require 160 bytes of storage in partition 1.

Note: Use the $STGUT1 utility (MX command) to monitor the number of interrupts buffered by the task supervisor.

INITMOD=

A list of entry point names in modules to be executed during system initialization. The maximum number of entries allowed in the user initialization list is 20. Each module specified on the INITMOD operand will require 2 bytes. For example, the INITMOD operand will require 20 bytes of storage in partition 1 for 10 initialization modules.

INITMOD=

(label1,label2,...,label20) The modules must be link-edited with the supervisor and cannot 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.

INITPRT= The partition into which $INITIAL is loaded after system initialization. The range for INITPRT is 0 to the maximum number of partitions supported by your processor. If you specify a partition without enough storage to contain $INITIAL, the system cannot load $INITIAL 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 in its search order (specified by $SRPROF) it finds with enough storage to contain $INITIAL.

LOGPART=

The number of the partition (1 to the maximum number of partitions supported by your processor) where the system will attempt to load $LOG first. The range for LOGPART is from 0 to the maximum number of partitions supported by your processor. If the system is unable to load $LOG in the selected partition, it displays an error message and attempts to load $LOG as if the default had been selected.

DEFAULT=

0. If the system is unable to load $LOG in the requested partition, it attempts to load $LOG starting with the maximum number of partitions supported. The system then moves downwards until it locates an available partition.

MECBLST=

The maximum number (in decimal) of multiple event control blocks (MECBs) allowed in the system at any given time. The range for MECBLST is 0 to 64. The system generates an MECB pointer consisting of two words (address and address key) for each MECB required. For example, if you specify 64, the system generates 128 words in partition 1. The MECB statement is used in conjunction with the WAITM instruction to permit a program to wait on the occurrence of multiple events.

TBPART = The number of the partition (1 to the maximum number of partitions supported) in which the unmapped storage table should be built. The range for TBPART is 0 to the maximum number of partitions supported by your processor. If the system is unable to construct the unmapped table in the partition, it displays an error message. If you use the default (0), the unmapped storage table will be built in the highest-numbered partition (starting with the maximum number of partitions supported and moving downwards) where there is sufficient storage available.

The table does not appear as a data area when you issue $A. Instead, the size of the table is subtracted from the size of the partition. $A shows the adjusted partition size.

VIRPART =

The number of the partition (1 to the maximum number of partitions supported) where the system first attempts to load $VIRLOG. The range for VIRPART is 0 to the maximum number of partitions supported. If the system is unable to load $VIRLOG in the selected partition, it displays an error message and attempts to load $VIRLOG as if the default had been selected.

DEFAULT =

0. If the system is unable to load $VIRLOG in the requested partition, it attempts to load $VIRLOG starting with partition the maximum number of partitions supported by your processor. The system then moves downwards until it locates an available partition.

XPSSTK = The number of entries in the cross-partition stack. Each entry in the cross partition stack requires 6 bytes of storage in partition 1. For example, if 50 entries are required, the stack will consist of 300 bytes. The range for XPSSTK is from 10 to 128. The default for XPSSTK is 20.

The stack contains the return address and partition numbers to which the supervisor refers when branching between partition 1 and supervisor portions in other partitions.

If your supervisor includes terminal I/O support outside of partition 1, a cross-partition stack entry is made for each terminal that is active. If the system does not have sufficient cross-partition stack entries, it will either stop or enter a continuous run loop. On systems with more than 10 terminals, you may want to specify an XPSSTK entry larger than the default of 20 to ensure that your system has sufficient cross-partition stack entries if you use an additional terminal.

Note: Use the $STGUT1 utility (MX command) to monitor the number of entries in the cross-partition stack. There may be instances when the stack size required is dependent on the number of terminals that the Series/1 is to support or instances where entries are placed on the stack even though all the supervisor resides in partition 1.

For examples of defining the SYSPARMS statement, see "Examples for the System Partition Statements" on page A-31.

SYSCOMM - Define Base and Common Partition

The SYSCOMM system partition statement defines which partition will be used as the base partition, which partitions will contain common storage, and the amount of common storage per partition (in 2K blocks). The SYSCOMM statement is optional, but if coded, it must follow the SYSPARTS and SYSPARMS statements and precede the SYSEND statement.

Common storage enables you to configure the system partitions to allow programs executing in one partition to access storage in another partition and in the current partition.

Note: A maximum of 254 characters are permitted in the operand field. Therefore, if all the operands in the SYSCOMM statement are coded, the 254 maximum characters might be exceeded and an error could occur. Avoid this error condition by allowing operands to default whenever possible. The number of characters in the COMMON operand field can be reduced by coding the number of 2K blocks, rather than labels, whenever possible.

Syntax:

blank	**SYSCOMM COMMON=,COMBASE=**
Required:	**COMMON=**
Defaults:	**COMBASE=1**

Operand ***Description***

COMMON=

The amount of storage (in 2K blocks) to map from the partition specified by COMBASE into each of the partitions specified in the SYSPARTS statement. A label or a number can be specified. The label specified must be located in partition 1. The address of the label will be rounded up to the next 2K boundary. For example, if the address of of the label was equal to 3.5K, the amount of COMMON defined for the partition corresponding to the label would be 4K. The number value represents the number of 2K storage blocks to map from the base partition. The range for a numeric operand is from 0 to the maximum number of partitions supported by your processor.

COMMON=nn

Where nn is the number of 2K blocks of storage to be mapped across all partitions defined by the NUMPART= operand on the SYSPARTS statement.

COMMON=label

The address of the label will be rounded up to the next 2K boundary. The 2K boundary value will be used to assign COMMON across all partitions defined by the NUMPART= operand on the SYSPARTS statement.

COMMON=(nn,...,nn,label1,label2,nn,nn,nn,nn,nn,nn,nn,nn)

Where each label or number defines the amount of COMMON to be mapped for that particular partition.

COMBASE=

The number of the partition (1 – 8) where COMMON is based for mapping. The default for COMBASE= is 1. The base partition contains the physical storage that other partitions access using the COMMON operand.

If you code a number other than 1, the first operand of the COMMON= keyword must be 0 or EDXSYS, and the partition into which common is mapped (the COMBASE partition) must be 0 or EDXSYS.

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 32) corresponds to address spaces 0 through 31. 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.

```
COMMON=(0,1,1,0,1)
```

The 0 is used as a place holder when you want to skip partitions. In the example above, the COMMON base partition (COMBASE) is 1. (See "Example 8" on page A-40 for an explanation of the use of 0).

To map the entire resident supervisor, specify COMMON=EDXSTART. To map only the supervisor data areas, specify COMMON=EDXSVCX. The default, COMMON=0, implies no mapping.

For examples of defining the SYSCOMM statement, see "Examples for the System Partition Statements" on page A-31.

SYSEND - Define End of System Partition Statements

The SYSEND system partition statement defines the end of the system partition statements. The SYSEND statement is required and must follow the other system partition statements.

Syntax:

blank	**SYSEND**
Required:	**None**

Operand	*Description*
None	Defines the end of the system generation area.

For examples of defining the SYSEND statement, see "Examples for the System Partition Statements" on page A-31.

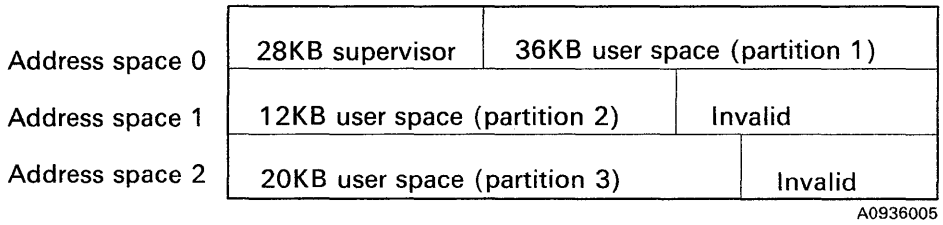
Examples for the System Partition Statements

Example 1: Define a three-partition system on a 96K-byte 4955.

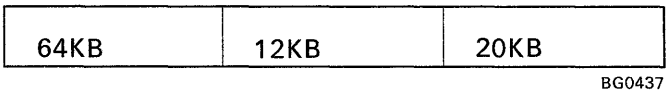
```
SYSPARTS    NUMPART=3,PARTS=(32,6,10),                    C
            MAXPROG=(3,2,3)
SYSEND
```

The system logically and physically maps in storage as follows:

Logical mapping:



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. 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 programs concurrently.
4. There is no unmapped storage.

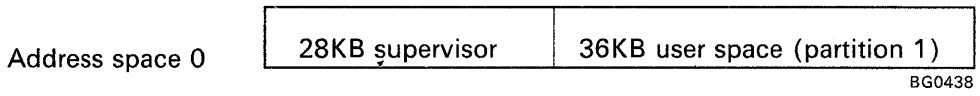
Note: The 28KB supervisor size is used for illustrative purposes only.

Example 2: Define a single-partition (64K) system.

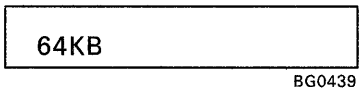
```
SYSPARTS   NUMPART=1,MAXPROG=5
SYSEND
```

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.

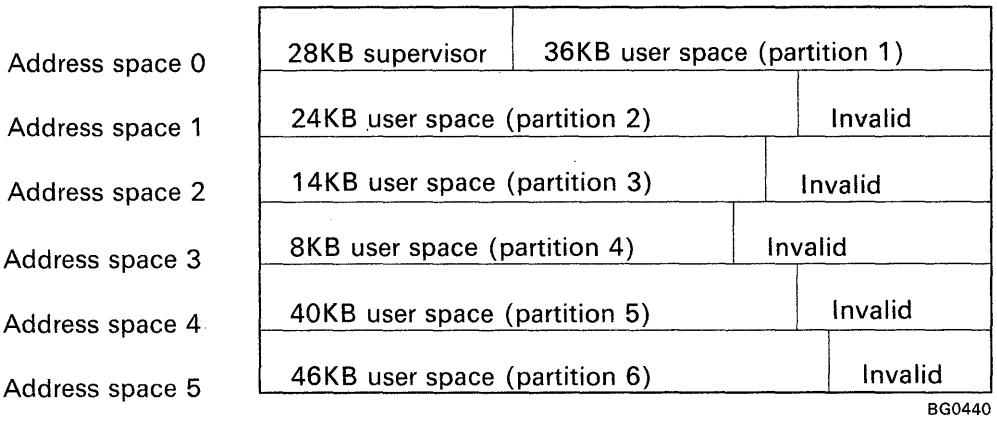
Note: The 28KB supervisor size is used for illustrative purposes only.

Example 3: Define a six-partition system on a 196K-byte 4955.

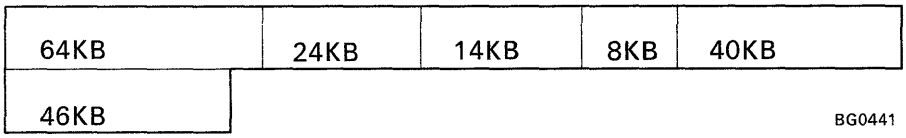
```
SYSPARTS    NUMPART=6,PARTS=(9,12,7,4,20,23),          C
            MAXPROG=(1,2,1,3,4,1)
SYSPARMS    INITPRT=2
SYSEND
```

The system logically and physically maps in storage as follows:

Logical mapping:



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 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.

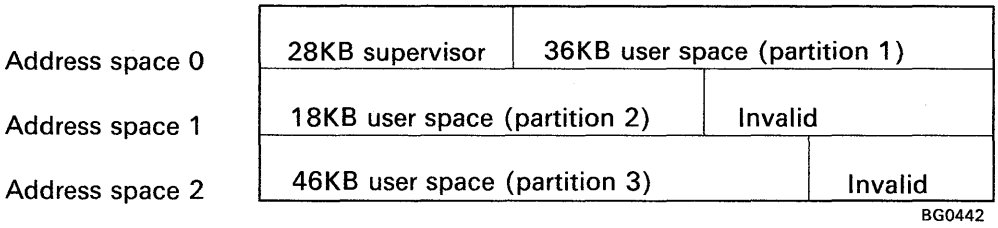
Note: The 28KB supervisor size is used for illustrative purposes only.

Example 4: Define a three-partition system on a 128K-byte 4955.

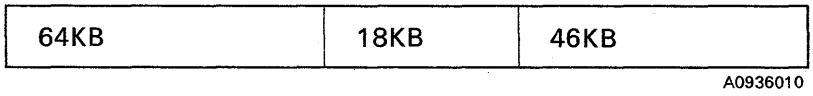
```
SYSPARTS   NUMPART=3,PARTS=(27,9,23),                C
           MAXPROG=(10,10,10)
SYSPARMS   INITMOD=(MYINIT)
SYSEND
```

The system logically and physically maps as follows:

Logical mapping:



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 ten programs concurrently.

   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 MYINIT will be activated during initialization.

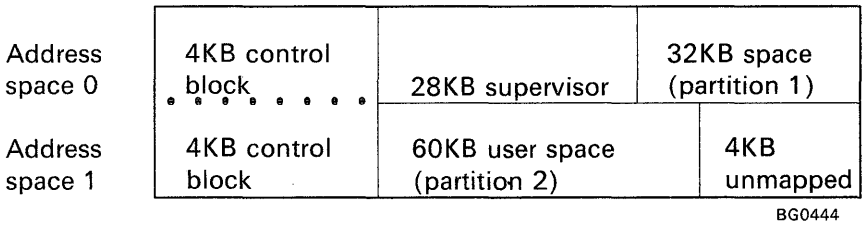
Note: The 28KB supervisor size is used for illustrative purposes only.

Example 5: Define a two-partition system on a 128K-byte 4952.

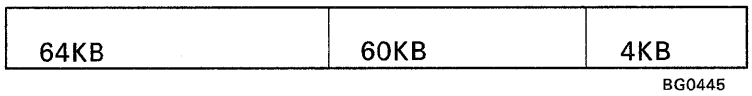
```
SYSPARTS    NUMPART=2,PARTS=(32,32),                    C
            MAXPROG=(3,6)
SYSPARMS    DATEFMT=MMDDYY
SYSCOMM     COMMON=EDXSVCX
SYSEND
```

The system logically and physically maps as follows:

Logical mapping:



Physical mapping:



1. Partition 1 has 32KB of supervisor space and 32KB of user space and can execute up to three programs concurrently.
2. Partition 2 has 4KB of common control blocks and 60KB of user space and can execute up to six programs concurrently.
3. Total storage used: 4KB + 28KB + 32KB + 60KB = 124KB. The size of processor storage is 128KB leaving 4KB of unmapped storage.

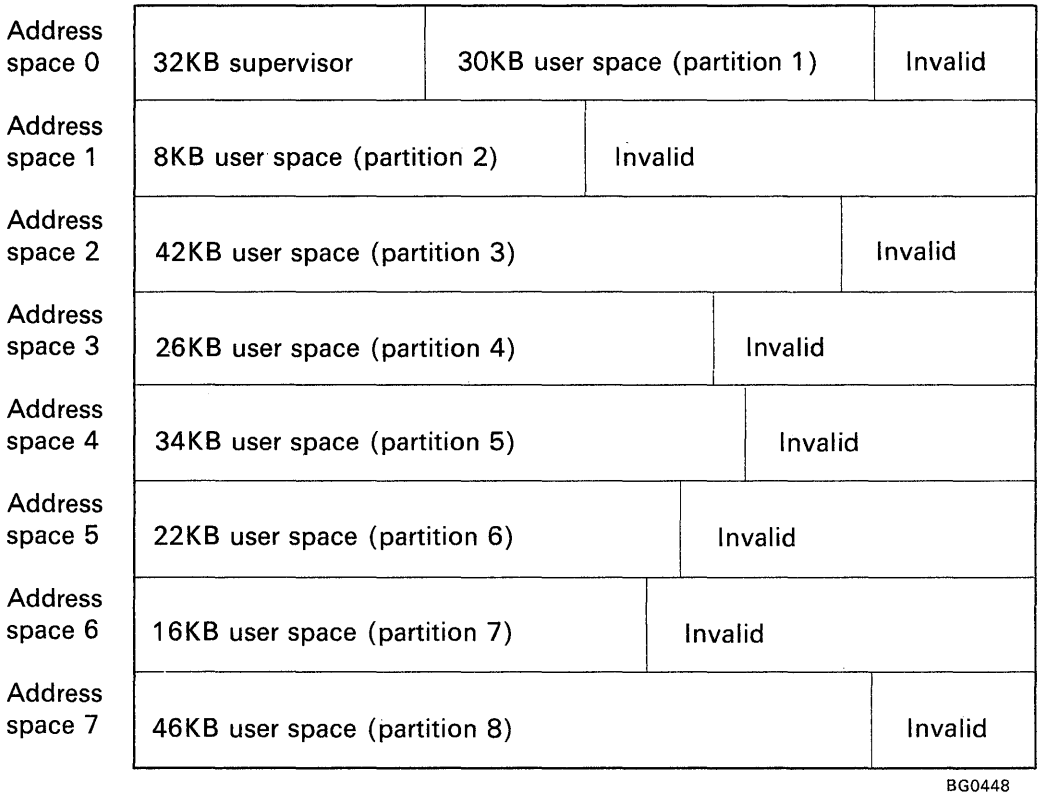
Note: The 28KB supervisor size and the 4KB control block size are used for illustrative purposes only.

Example 6: Define an eight-partition system on a 256K-byte 4955.

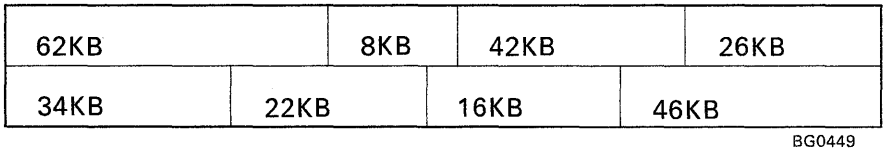
```
SYSPARTS   NUMPART=8,PARTS=(15,4,21,13,17,11,8,23),      C
           MAXPROG=(3,1,5,2,2,1,1,4)
SYSEND
```

The system logically and physically maps as follows:

Logical mapping:



Physical mapping:



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.

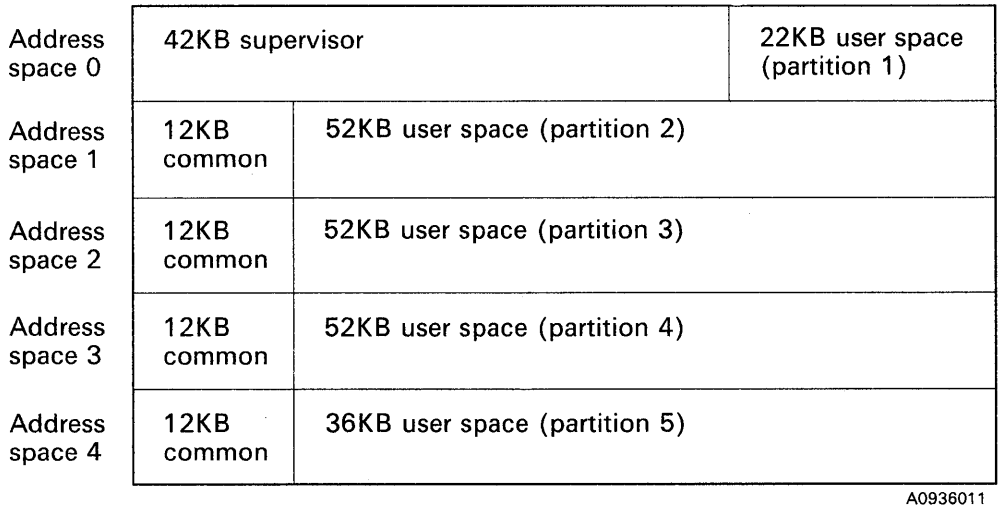
Example 7: Define a five-partition system on a 256KB 4955.

```
SYSPARTS   NUMPART=5,PART=(26,26,26,26,26),                C
           MAXPROG=(10,10,10,10,10)
SYSCOMM    COMMON=USRMAP
SYSEND
```

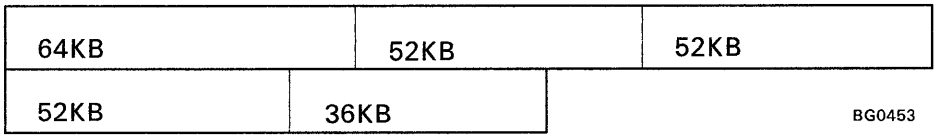
The supervisor requires 42K bytes of storage and the COMMON area requires 12K bytes of storage. USRMAP is the label of an entry point in the user module set up as a COMMON area. In the above example, USRMAP is located at address X'3000'. This address indicates the end of the COMMON area, which is 12K bytes in length.

The system logically and physically maps as follows:

Logical mapping:



Physical mapping:



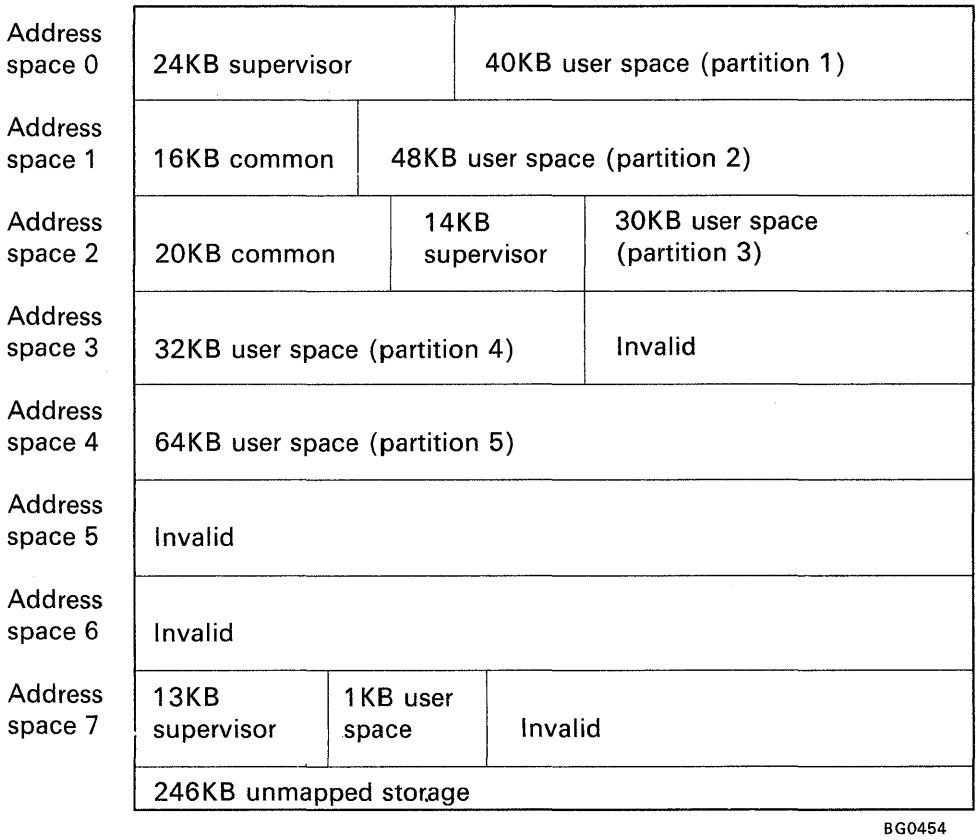
1. Partition 1 requires 21 blocks (42KB) for the supervisor and has 11 blocks (22KB) left for user space. Partition 1 can execute up to 10 programs concurrently.
2. Partition 2 requires 6 blocks (12KB) to serve as addressing pointers to the COMMON area and has 26 blocks (52KB of actual storage mapped 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.

Example 8: Define a five-partition system on a 512K-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.

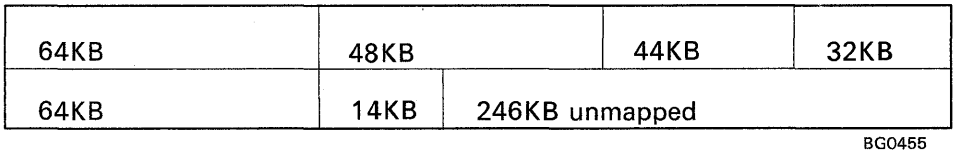
```
SYSPARTS   NUMPART=5,PARTS=(32,32,32,16,32),          C
           MAXPROG=(1,5,5,5,5)
SYSCOMM    COMMON=(0,8,10)
SYSEND
```

The system logically and physically maps as follows:

Logical mapping:



Physical mapping:



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. The 0 (zero) is used as a place holder in the COMMON= operand. Since you cannot have a COMMON area in partition 1 if you want multiple COMMON areas, you have to begin with a 0. If you want to skip partitions, you also use 0 to indicate this.
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 requires 32 blocks (64KB) of storage for user space. Partition 5 can execute up to 5 programs concurrently.
6. Partitions 6 and 7 are not defined.
7. Partition 8 was not defined on the SYSPARTS 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 processor storage is 512KB. Only 266KB of storage was defined leaving 246KB of unmapped storage.

Example 9: Define a 24-partition system on a 2048K-byte 4956. A single partition supervisor was generated for this system with 50K of supervisor code in partition 1.

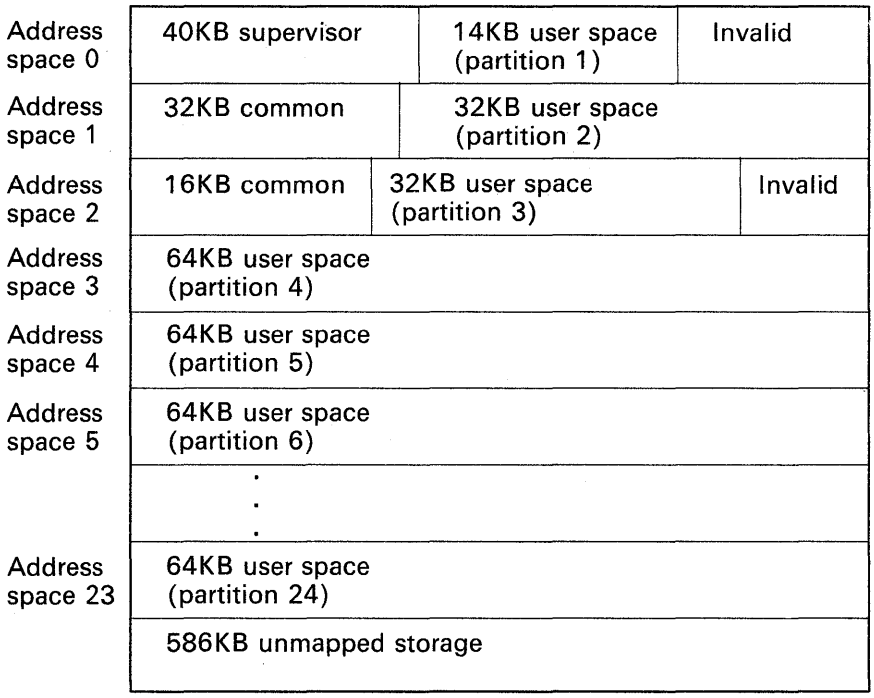
```
SYSPARTS    NUMPART=24,PARTS=(2,16,16),                C
            MAXPROG=(1,10,1)
SYSCOMM     COMMON=(0,LABEL,8)
SYSEND
```

If you want to include data in the supervisor to be used as common data, you must include a module in the supervisor with your data. This module must have a label (EXTRN) at the end, specified on the COMMON= operand. The (0,LABEL,8) in this example indicates that you want the area up to LABEL set aside in partition 2. You can include LABEL in partition 1 only. In this case, LABEL is at address X'8000' of the supervisor in partition 1.

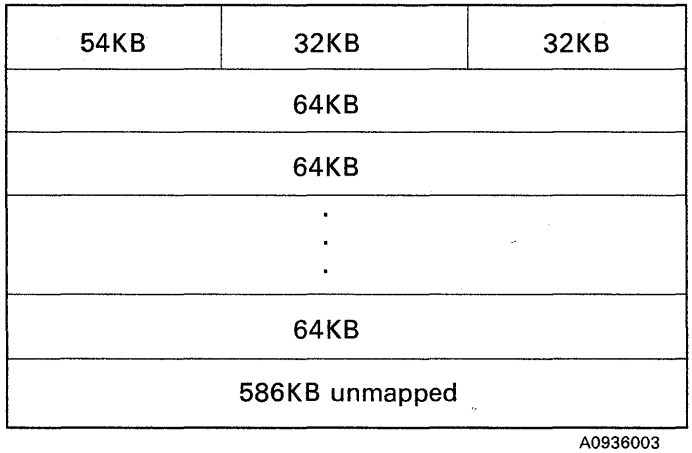
If the label comes first, as in COMMON=LABEL, you indicate that you want the COMMON area mapped in all partitions. Then all the COMMON areas will have the name of the label.

The system logically and physically maps as follows:

Logical mapping:



Physical mapping:



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. Partitions 4 – 24 are 64KB of user space.
5. The size of process storage is 2048KB. Only 1462KB of storage was defined leaving 586KB 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.
ID=	A 1- to 6-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

The TERMINAL statement 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, SYSPARTS, 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 A-1.

Terminal	Feature Number	DEVICE Operand of TERMINAL	CODTYPE Operand
IBM 3101 terminal, all models. Also 3151/3161/3163/3164 terminals.	#1310, #2095/#2096, 2095/RPQ D02350	ACCA	ASCII
IBM 3101 terminal, all models. Also 3161/3163/3164 terminals.	#1610, #2091/#2092	ACCA	EBASC
IBM 3101 terminal, models 10, 12, and 13. Also 3161/3163/3164 terminals in 3101 emulation mode only.	#7850	TTY[1]	ASCII
IBM 4978 terminal	RPQ D02038	4978	N/A
IBM 4979 terminal	#3585	4979	N/A
IBM 4980 terminal	#1250	4980	N/A
Teletype[1] ASR 33/35 (TTY) or equivalent	#7850	TTY	ASCII
ASCII terminal	#2095/#2096	ACCA	ASCII
ASCII terminal	#1610, #2091/#2092	ACCA	EBASC
IBM 2741 Communication Terminal	#1610	2741	CRSP or EBCD

Figure A-1 (Part 1 of 2). Supported EDX Terminals and DEVICE/CODTYPE Operands

[1] Trademark of Teletype Corporation

Terminal	Feature Number	DEVICE Operand of TERMINAL	CODTYPE Operand
IBM 4973 printer	#5630	4973	N/A
IBM 4974 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, #2095/2096	ACCA	ASCII
IBM 5219 printer	#5640	5219	N/A
IBM 5224 printer	#5640	5224	N/A
IBM 4234, 5225 and 5262 printers	#5640	5225	N/A
IBM 4201 and 4202 printers	#1310, #2095/#2096	4201	ASCII
IBM 4224 printer	#1310, #2095/#2096	4224	ASCII, EBCDIC, or user-defined transmission code
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 A-1 (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. (Refer to the *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, and #2095/#2096.

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-A 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; refer to 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 4980 terminal. $SYSLOG is the primary device to receive/display all exception messages. When $VIRLOG (DISK = YES or CF = YES) is active, it renames the $SYSLOG device to #SYSLOG.

$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.

$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 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 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 table, the left-hand column lists the operands available for the TERMINAL statement. The heading of each column lists a different device. An "X" indicates the operands that you can specify for that particular device (as determined by the DEVICE operand of the TERMINAL statement).

The sections following the tables describe the syntax and operands for each supported device by device type.

Notes:

1. You must group together all TERMINAL statements and include an END=YES for the last statement.
2. Specify TYPE=DSECT on the TERMINAL statement to include the EDL copy code when assembling under $S1ASM and using the Macro Library. (TYPE=CSECT is the default.)
3. You must have unique labels on TERMINAL statements, tape IDs (see the TAPE definition statement), and volume names (see the DISK definition statement). 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, it requires two virtual terminals. (See "Examples and Defaults" on page A-112 for a sample configuration.) (For a detailed description of the PASSTHRU function, refer to the Remote Management Utility chapter in the *Communications Guide*.)

Device-dependent operands for the TERMINAL statement:

	2741	4013	4973	4974	4975	4975-01A	5219
ADAPTER	X				X	X	X
ADDRESS	X		X	X	X	X	X
ATTN						X	
BITRATE	X				X	X	
BOTM	X	X	X	X	X	X	X
CHARDEL	X	X				X	
CHARSET					X		X
CKSUM							
COD							
CODTYPE	X	X					
CR	X	X					
CRDELAY	X	X					
DI/DO/PI		X					
ECHO	X	X					
END	X	X	X	X	X	X	X
HDCOPY							
LEFTM	X	X	X	X	X	X	X
LF	X	X					
LINEDEL	X	X					
LINSIZE	X	X	X	X	X	X	X
LMODE					X		
MODE							
NHIST							
OVFLINE	X	X	X	X	X	X	X
PAGSIZE	X	X	X	X	X	X	X
PART	X	X					
PACING							
PF1							
PORT							
RANGE							
RIGHTM	X	X	X	X	X	X	X
SCREEN	X	X					

	2741	4013	4973	4974	4975	4975-01A	5219
SECADDR							
SHARING							
SPOOL	X		X	X	X	X	X
SYNC							
TIMERS							
TOPM	X	X	X	X	X	X	X
TYPE	X	X	X	X	X	X	X

Device-dependent operands for the TERMINAL statement (continued):

	5225	4234/5262	5224	4978	4979	4980	ACCA
ADAPTER	X	X			X	X	X
ADDRESS	X	X	X	X	X	X	X
ATTN				X	X	X	X
BITRATE						X	X
BOTM	X	X	X	X	X	X	X
CHARDEL							X
CHARSET	X	X	X				
CKSUM							
COD							X
CODTYPE							X
CR							X
CRDELAY							X
DI/DO/PI							
ECHO							
END	X	X	X	X	X	X	X
HDCOPY				X	X	X	
LEFTM	X	X	X	X	X	X	X
LF	X	X					
LINEDEL							X
LINSIZE	X	X	X	X	X	X	X
LMODE							X
MODE							X
NHIST				X	X	X	X
OVFLINE	X	X	X	X	X	X	X
PAGSIZE	X	X	X				X
PART	X	X	X				
PACING							X
PF1				X	X	X	X
PORT	X	X	X			X	
RANGE							X
RIGHTM	X	X	X	X	X	X	X
SCREEN				X	X	X	X

	5225	4234/5262	5224	4978	4979	4980	ACCA
SECADDR	X	X	X			X	
SHARING							X
SPOOL	X	X	X	X	X	X	X
SYNC							
TIMERS							X
TOPM	X	X	X	X	X	X	X
TYPE	X	X	X	X	X	X	X

Device-dependent operands for the TERMINAL statement (continued):

	4201/4202	4224	TTY	PROC	VIRT	GPIB	S1S1
ADAPTER	X	X					
ADDRESS	X	X	X	X	X	X	X
ATTN			X				X
BITRATE	X	X		X			
BOTM	X	X	X				
CHARDEL			X				
CHARSET	X	X					
CKSUM							X
COD							
CODTYPE	X	X	X	X		X	
CR			X	X			
CRDELAY			X	X			
DI/DO/PI							
ECHO			X				
END	X	X	X	X	X	X	X
HDCOPY							
LEFTM	X	X	X				
LF			X	X			
LINEDEL			X				
LINSIZE	X	X	X	X	X	X	X
LMODE	X	X					
MODE	X	X	X				
NHIST							
OVFLINE	X	X	X				
PAGSIZE	X	X	X				
PART			X			X	
PACING							
PF1			X				
PORT							
RANGE	X	X		X			
RIGHTM	X	X	X				
SCREEN			X			X	

	4201/4202	4224	TTY	PROC	VIRT	GPIB	S1S1
SECADDR							
SHARING	X						
SPOOL	X	X	X			X	
SYNC					X		
TIMERS	X	X					
TOPM	X	X	X				
TYPE	X	X	X	X	X	X	X

2741 Terminal

A TERMINAL definition statement with DEVICE=2741 defines a 2741 communications terminal attached through the 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 through the 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.

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 that 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 that 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 the *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.

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 that ends 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 with which the terminal is normally associated. Code 1–32 for the 4956 models J and K. Code 1–16 for the 4956 models E, 60E, and H (with extended address mode defined) or 1–8 for all others. 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 an 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=A0,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 through the 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 through the 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.
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, see "ASCII Transmission Codes" on page A-47.

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 operand 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, see "ASCII Transmission Codes" on page A-47.

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 ends 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, see "ASCII Transmission Codes" on page A-47.

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 with which the terminal is normally associated. Code 1 − 32 for the 4956 models J and K. Code 1 − 16 for the 4956 models E, 60E, and H (with extended address mode defined) or 1 − 8 for all others. The default is partition 1.

You can change the partition assignment at execution time with the $CP operator command as described in the *Operation Guide*.

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 = 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
```

4201/4202/4224 Printer

A TERMINAL definition statement with DEVICE=4224 defines a 4224 printer connected through a Multifunction Attachment Feature (MFA) #1310 or a #2095 with #2096. A TERMINAL definition statement with DEVICE=4201 defines a 4201 or 4202 printer connected through a Multifunction Attachment Feature #1310 (MFA) or a #2095 with #2096.

See Appendix F, "Setting Up the 4201, 4202, and 4224 Printers" on page F-1 for information on setting up your 4224, 4201, or 4202 printer.

Note: References to the 4201 printer throughout the EDX Library also apply to the 4202 printer unless otherwise noted. This section contains additional information on the 4202 printer.

Syntax:

label	**TERMINAL**	**DEVICE=,ADDRESS=,MODE=,PAGSIZE=, LINSIZE=,CODTYPE=,TOPM=, BOTM=,LEFTM=,RIGHTM=,OVFLINE=, BITRATE=,RANGE=,LMODE=,ADAPTER=, TIMERS=,SPOOL=,SHARING=,END=**
Required:	**DEVICE=,ADDRESS=,ADAPTER=**	
Defaults:	**LINSIZE=132 (for 4224),LINSIZE=80 (for 4201 and 4202), MODE=STANDARD,CODTYPE=ASCII,PAGSIZE=66, TOPM=3,BOTM=62,LEFTM=0,RIGHTM=131 (FOR 4224), RIGHTM=79 (for 4201 and 4202),OVFLINE=NO, BITRATE=300 (1200 for MFA),RANGE=HIGH, LMODE=LOCAL,TIMERS=(1,1,1,1),SPOOL=YES, SHARING=NO,END=NO**	

label Required if ADAPTER=MFA; otherwise optional.

Operand	*Description*	
DEVICE=	One of the following device types:	
	4224	4224 ACCA printer attached through the 1310 (MFA) or a 2095 controller with a 2096 adapter.
	4201	4201 or 4202 ACCA printer attached through the 1310 (MFA) or a 2095 controller with a 2096 adapter.
ADDRESS=	The address (in hexadecimal) of the device.	
MODE=	Specify STANDARD to define the printer as a line printer. Specify MODE=PAGE to define the printer as a page printer. The default is MODE=STANDARD.	

In STANDARD mode, when printer output crosses a logical page boundary, the printer support issues line feeds to start a new page and then issues additional line feeds to position the paper on the correct line. In PAGE mode, when printer output crosses a logical page boundary, the printer support issues a form feed to start a new page and then issues line feeds to position the paper on the correct line.

For more information, refer to the section "Additional Printer Information" under the 4224 TERMCTRL description, or the "Special Considerations" section under the 4201 TERMCTRL description in the *Language Reference*.

MODE=VERIFY activates the verification feature for the 4224 printer which continually checks that the 4224 printer is still powered up. When this feature is active, the 4224 support issues a status request at the beginning of each line and then waits for a response from the printer.

If the 4224 is powered on, it typically takes from 1 to 10 milliseconds to respond. However, when crossing a page boundary, the 4224 may take several seconds to respond because it must print several lines of buffered data to complete the page.

If the 4224 support gets no response after 30 seconds, the output operation terminates with a return code of 96, indicating that the 4224 printer is not responding. Otherwise, the output operation proceeds normally.

This option is valid **only** for the 4224 printer attached locally.

MODE=VERPAGE is a combination of MODE=PAGE and MODE=VERIFY. This option is valid **ONLY** for the 4224 printer attached locally.

PAGSIZE= The logical page size (number of lines per page) of the printer. Specify a decimal number between 1 and 255. The default is PAGSIZE=66.

LINSIZE= The maximum length of an output line for the printer. 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 4224 printer and LINSIZE=80 for the 4201 and 4202 printers.

The system buffer in the CCB is always set to 230 to allow for TERMCTRL control sequences that the printer support generates. The maximum value for LINSIZE is 200 to allow for TERMCTRL control sequences. Refer to the *Language Reference* for more information.

The maximum 4224 print line is 13.2 inches. The maximum number of characters that can fit on a physical line depends on the density as follows:

Density	Maximum Line Size
Normal (12 CPI)	158
Compressed (15 CPI)	198
Expanded (10 CPI)	132

The maximum 4201 print line is 8 inches. The maximum number of characters that can fit on a physical line depends on the density as follows:

Density	Maximum Line Size
Normal (12 CPI)	96
Compressed (17.1 CPI)	136
Expanded (10 CPI)	80

The maximum 4202 print line is 13.6 inches. The maximum number of characters that can fit on a physical line depends on the density as follows:

Density	Maximum Line Size
Normal (12 CPI)	163
Compressed (17.1 CPI)	200 (software limit)
Expanded (10 CPI)	136

CODTYPE= The transmission code used by the terminal. Specify either ACSII, EBCDIC, or a user-defined transmission code (no more than six characters). If you specify EBCDIC, no translation table is used. Otherwise, the system appends the characters "TR" to the name you specify and assumes there is a translation table with that name.

Since the 4201 and 4202 are ASCII printers, you can specify CODTYPE=EBCDIC ***only*** if the text being transmitted is in ASCII already.

TOPM= The top margin (a decimal number between zero and PAGSIZE−1) to indicate the first line of the page which is to contain the printed data. For example, if TOPM=3 (the default), lines 0, 1, and 2 would be blank and line 3 would be the first printed line.

BOTM= The bottom margin, the last usable line on a page. Its value must be between TOPM and PAGSIZE−1. If an output instruction would cause the line number to increase beyond this value, then a page eject is performed before the operation is continued. For example, if BOT=62 (the default) and PAGSIZE=66, then line 62 would be the last printed line and lines 63, 64, and 65 would be blank.

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=LINSIZE−1.

OVFLINE= YES, if output data which exceeds the right margin is to be continued on the next line. NO if output data which exceeds the right margin is to be truncated. 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.

OVFLINE support detects line overflow based on the allowable line size which is computed from the left and right margins. The line size is calculated as a character count, ***not*** as a measurement in inches. When the number of characters exceeds the allowable line size, the rest of the line is either truncated or continued on the next physical line as specified by the OVFLINE operand. Since the 4224 printer allows such a wide variety of print modes and print styles, you must make sure that a single *logical* line can fit on a *physical* line.

BITRATE= The rate (in bits per second) that this terminal will be operating. For the MFA adapter, you can specify 1200, 2400, 4800, or 9600. For the #2095/#2096 adapter, the RANGE= operand determines the allowable bit rates. Make certain that the bit rate you specify here matches the bit rate selected on the printer.

For the 4201 and 4202 printers attached to the MFA adapter, 4800 is the maximum bit rate supported.

RANGE= Specify HIGH or LOW to match the hardware jumper that is installed on the adapter card.

RANGE must be HIGH for the MFA adapter.

If RANGE=LOW, the bit rate can be anywhere from 38 to 1200. If RANGE=HIGH, the bit rate can be anywhere from 300 to 19200.

LMODE= The following options are available:

RS422 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, through the RS-232-C port.

SWITCHED For a connection that is switched. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only. Request-to-Send is not permanently activated.

SWPRTS For a connection that is switched. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only. It permanently activates Request-to-Send (RTS). This is equivalent to installing the RTS jumper on a single-line adapter.

PTTOPT For a connection that is point-to-point nonswitched. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only. Request-to-Send is not permanently activated.

PTTPRTS For a connection that is point-to-point nonswitched. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only. It permanently activates Request-to-Send (RTS). This is equivalent to installing the RTS jumper on a single-line adapter.

The LMODE specified should match the jumpers on the controller cards. The Multifunction Attachment does not have jumpers. (See Appendix B, "Customizing Adapters with Hardware Jumpers" for hardware considerations.)

ADAPTER = Code one of the following:

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.

The following list illustrates the various combinations of ADAPTER and LMODE for the 4224, 4201, or 4202 printer:

ADAPTER = MFA,LMODE = LOCAL: Specifies a connection with the #1310 (MFA) attachment through the only available RS-232-C port.

ADAPTER = MFA,LMODE = RS422: Specifies a connection with the #1310 (MFA) attachment through one of the four RS-422-A ports.

ADAPTER = TWO or FOUR: Specifies a connection with the #2095/2096 Feature Programmable Attachment with a single 2096 card using the RS-232-C interface.

ADAPTER = SIX or EIGHT: Specifies a connection with the #2095/2096 Feature Programmable Attachment with two 2096 cards using the RS-232-C interface.

TIMERS = The T1 and T2 values for receive and transmit operations in a sublist format. For example, TIMERS = (4,5,6,7) specifies:

Receive T1 = 4

Receive T2 = 5

Transmit T1 = 6

Transmit T2 = 7

Unless you suspect hardware errors, you should allow TIMERS = to default to (1,1,1,1).

SPOOL = YES (the default), to define the printer as a valid spoolable device.

NO, to specify that the printer is not a valid spoolable device. However, you can redefine this device as a spoolable device at a later time using $TERMUT1.

SHARING = YES, to specify this device as the printer attached to a line sharing primary device. The system sends data to the printer on the same physical line as the primary device.

NO (the default), inhibits the use of line sharing with this device.

Line sharing is valid only for the Multifunction Attachment Feature (#1310) and the Feature Programmable Attachment (#2095/2096).

The TERMINAL statement for the attached printer must immediately follow the TERMINAL statement for the primary device. See the SHARING = parameter of the ACCA TERMINAL statement for information on the primary device.

You must code DEVICE = 4201 to define the printer as a 4201/4202. The device address must be the same as the device address for the primary device.

If the shared line is attached to the Multifunction Attachment Feature (#1310), the ADAPTER parameter for the printer must be coded as ADAPTER=TWO (not ADAPTER=MFA). The label of the TERMINAL statement for the printer must not appear in the ADAPTER statement for the Multifunction Attachment Feature (#1310).

For the #2095/#2096 attachment, code ADAPTER= to match the value coded for the primary device.

ATTN=NO must not be specified.

If you are connecting an IWS Personal Computer work station to a #2095/#2096 attachment card, specify a transmit T1 value of at least 3 on the 4201/4202 TERMINAL statement. For example, TIMERS=(1,1,3,1). The attachment card uses the transmit T1 value, multiplied by 3.33 milliseconds, for the pretransmit delay.

The BITRATE= parameter for the attached printer is automatically set to the BITRATE specified for the primary device because this is the rate at which all data (including printer data) travel between the Series/1 and the primary device. The maximum rate for the Multifunction Attachment Feature (#1310) is 9600 BPS. The maximum bit rate for the Feature Programmable Attachment (#2095/2096) is 19,200 BPS.

If the shared line is attached to an RS-422-A port of the MFA feature #1310, code LMODE=LOCAL for the printer. Otherwise, code LMODE= to match the value code for the primary device.

For a 4201/4202 printer attached to a 3151, 3161, 3163, or 3164 terminal, you must set the software switches for the AUXILIARY PORT of the terminal. PARITY must be set to NO. STOP BIT must be set to 2. WORD LENGTH must be set to 8. LINE SPEED must be set to match the line speed selected on the serial attachment of the 4201/4202 printer. A line speed of 2400 BPS should be adequate. A line speed in excess of 4800 may cause printer buffer overrun. See "Software Setup Switches for the 3151, 3161, 3163, and 3164" on page C-7 for information on how to set these switches.

To get 4201 TERMCTRL support, you must include module IOS4224 in the Terminal I/O partition.

You must include modules IOSACCA and IOS4975A for 4201/4202 printer support.

Refer to the "Line Sharing and XON/XOFF Pacing" chapter in the *Communications Guide* for more information on line sharing.

END= YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

You cannot change the values for the following operands (they are set as shown):

- CRDELAY=00
- CR=0D
- LF=0A
- COD=(13).

Examples and Defaults

Example 1: The following example contains sample values for defining a 4224 printer when it is connected through the Multifunction Attachment Feature (#1310).

```
label     TERMINAL  DEVICE=4224,ADDRESS=58,ADAPTER=MFA

is equivalent to:

label     TERMINAL  DEVICE=4224,ADDRESS=58,PAGSIZE=66,          C
                    LINSIZE=132,CODTYPE=ASCII,TOPM=3,BOTM=62,   C
                    LEFTM=0,RIGHTM=131,OVFLINE=NO,BITRATE=9600, C
                    RANGE=HIGH,LMODE=LOCAL,ADAPTER=MFA,         C
                    TIMERS=(1,1,1,1),SPOOL=YES,END=NO
```

Example 2: The following example contains sample values for defining a 4201 printer when it is connected through the Feature Programmable Attachment (#2095/2096).

```
label     TERMINAL  DEVICE=4201,ADDRESS=68,ADAPTER=FOUR

is equivalent to:

label     TERMINAL  DEVICE=4201,ADDRESS=68,PAGSIZE=66,          C
                    LINSIZE=80,CODTYPE=ASCII,TOPM=3,BOTM=62,    C
                    LEFTM=0,RIGHTM=79,OVFLINE=NO,BITRATE=300,   C
                    RANGE=HIGH,LMODE=LOCAL,ADAPTER=FOUR,        C
                    TIMERS=(1,1,1,1),SPOOL=YES,END=NO
```

Example 3: The following example contains sample values for defining a 4202 printer when it is connected through the Feature Programmable Attachment (#2095/2096).

```
label     TERMINAL  DEVICE=4201,ADDRESS=68,ADAPTER=FOUR

is equivalent to:

label     TERMINAL  DEVICE=4201,ADDRESS=68,PAGSIZE=66,          C
                    LINSIZE=80,CODTYPE=ASCII,TOPM=3,BOTM=62,    C
                    LEFTM=0,RIGHTM=79,OVFLINE=NO,BITRATE=300,   C
                    RANGE=HIGH,LMODE=LOCAL,ADAPTER=FOUR,        C
                    TIMERS=(1,1,1,1),SPOOL=YES,END=NO
```

Example 4: The following example contains sample values for defining a 4201 printer when it is connected through a shared line to an RS-422-A port of the Multifunction Attachment Feature (#1310).

```
label     TERMINAL  DEVICE=4201,ADDRESS=5B,                    C
                    LINSIZE=132,CODTYPE=ASCII,                 C
                    BITRATE=9600,                              C
                    LMODE=LOCAL,ADAPTER=TWO,                   C
                    SHARING=YES,END=NO
```

Example 5: The following example contains sample values for defining a 4201 printer connected through a shared line to the Feature Programmable Attachment (#2095/2096).

```
label     TERMINAL  DEVICE=4201,ADDRESS=AB,                    C
                    LINSIZE=132,CODTYPE=ASCII,                 C
                    BITRATE=9600,                              C
                    LMODE=LOCAL,ADAPTER=EIGHT,                 C
                    SHARING=YES,END=NO
```

4973/4974 Printers

A TERMINAL definition statement with DEVICE=4973 (or 4974) defines a 4973 printer attached through the 5630 adapter or a 4974 printer attached through the 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 printer attached through the 5630 Adapter **4974** 4974 printer attached through the 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.

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 the terminal as a valid spoolable device.

NO, to specify the terminal as 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 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 printer (connected locally or remotely) through a Mutlifunction Attachment Feature #1310 (MFA).

Note: For the 4975-01A printer, see the section "ACCA-Type Terminals" on page A-89.

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=62,LEFTM=0,RIGHTM=131,OVFLINE=NO, BITRATE=1200,LMODE=LOCAL,ADAPTER=MFA, 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 printer locally attached through the #1310 (MFA) Adapter.
	4975-01R or -02R 4975 printer remotely attached through the #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, refer to the PDEN= operand of the TERMCTRL instruction in the *Language Reference*.

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 and the print mode for the 4975 are set automatically 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=62.

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.

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 operand 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-C) support connected through the #1310 attachment, there are three options: LOCAL, PTTOPT and PTTPRTS. The default is LMODE=LOCAL.

- **LOCAL** Means that a 4975 R model is directly connected to a Series/1 with no modems. This is the default.
- **PTTOPT** Means a 4975 R model is connected to a Series/1 on a leased line with internal (1200 BPS) or external (1200, 2400, or 4800 BPS) modems.
- **PTTPRTS** Provides the same function as PTTOPT with one exception: PTTPRTS permanently activates Request-to-Send (RTS).

This operand is not valid for 4975 L (RS-422-A) 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 the terminal as a valid spoolable device.

NO, to specify the terminal as 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 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 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 or PTTPRTS. In either 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 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= is always LOCAL and BITRATE= is 2400 for the 4975-01L and 4800 for the 4975-02L.

4978/4979 Display Terminals

A TERMINAL definition statement with DEVICE=4978 (or 4979) defines a 4978 terminal attached through an RPQ D02038 or a 4979 terminal attached through the 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,PART=1,SPOOL=NO,END=NO**

Operand *Description*

DEVICE= One of the following device types:

4979 4979 terminal attached through the 3585 adapter

4978 4978 terminal attached through an 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 terminal), 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.

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. If you do not code this operand, the terminal will not use the hard-copy function.

```
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 will activate 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).

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= operand 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 terminal, 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.*

PART= The partition with which the terminal is normally associated. Code 1–32 for the 4956 models J and K. Code 1–16 for the 4956 models E, 60E, and H (with Extended Address Mode defined) or 1–8 for all others. 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 the terminal as a valid spoolable device.

NO (the default), to specify 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.

Example and Defaults

In the following example, the default assignments for the 4978/4979 terminal 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 terminal TERMINAL statement.

```
label    TERMINAL  DEVICE=4978,ADDRESS=04,HDCOPY=$SYSPRTR

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 Terminal

A TERMINAL definition statement with DEVICE=4980 defines a 4980 terminal attached through 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,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 terminal.
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 that follows.) The default is NHIST=12. **Note:** This operand is meaningful for roll screens only.
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. Refer to 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. If you do not code this operand, the terminal will not use the hard-copy function.

```
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 will activate 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.

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= operand 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 terminal, 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.

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 with which the terminal is normally associated. Code 1−32 for the 4956 models J and K. Code 1−16 for the 4956 models E, 60E, and H (with Extended Address Mode defined) or 1−8 for all others. 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 the terminal as a spoolable device.

NO (the default), to specify the terminal as 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). Refer to 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.

Example and Defaults

In the following example, the default assignments for the 4980 terminal 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 TERMINAL statement.

```
label    TERMINAL  DEVICE=4980,ADDRESS=80,ADAPTER=SMIO,          C
               PORT=0,SECADDR=01,HDCOPY=$SYSPRTR
```

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
```

4234/5219/5224/5225/5262 Printers

A TERMINAL definition statement with DEVICE=5219/5224/5225 defines one of the following printers:

- 5219 printer
- 5224 printer
- 4234, 5225, or 5262 printer.

connected to the Series/1 through 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=,PORT=,SECADDR=**
Defaults:		**PAGSIZE=66,LINSIZE=132,CHARSET=USCA,TOPM=3, BOTM=63,LEFTM=0,RIGHTM=131,OVFLINE=NO, SPOOL=YES,END=NO,ADAPTER=ALPA,END=NO**

label Required

Operand	*Description*
DEVICE=	One of the following device types: **5219** A 5219 printer locally attached through the Printer Attachment - 5200 Series **5224** A 5224 printer locally attached through the Printer Attachment - 5200 Series **5225** A 4234, 5225, or 5262 printer locally attached through 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.

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 characters 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.

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 the terminal as a valid spoolable device.

NO, to specify the terminal as 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=NO.

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
```

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,        C
                   SPOOL=YES,PORT=0,SECADDR=06,END=YES
```

Example 3: 4234 or 5262 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,              C
               SPOOL=YES,PORT=0,SECADDR=06,END=YES
```

Example 4: 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
```

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 or equivalent devices, these adapters support both locally-attached and remote terminals. The 3101 equivalent devices referred to in this section include the 3151, 3161, 3163, and 3164 terminals operating in either block or character mode.

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=, SPOOL=,PACING=,SHARING=,END=**
Required:	**DEVICE=,ADDRESS=**	
Defaults:	**PART=1,END=NO,OVFLINE=NO,ATTN=YES,SPOOL=NO, ADAPTER=SINGLE,SHARING=NO,PACING=NO**	

label Required if ADAPTER=MFA; otherwise optional.

Operand	*Description*
DEVICE=	ACCA, for an ASCII terminal attached through a 1610 controller, a 2091 controller with a 2092 adapter, a 2095 controller with a 2096 adapter, or a 4975-01A printer attached through a #2095/2096 or a 1310 (MFA). ACCA, for a 3101 terminal or an equivalent device in either block or character mode attached through any of the methods above. ACCA, for the Remote Support Link. (See "Examples and Defaults" on page A-98 for examples.)
ADDRESS=	The address (in hexadecimal) of the device.
MODE=	3101B, for a 3101 terminal or equivalent device that operates in 3101 block mode. 3101C, for a 3101 terminal or equivalent device that operates in 3101 character mode. 3161B for a 3151 terminal in 3161 mode. 3161B for a 3161 terminal in 3161 mode. 3163B for a 3163 terminal in 3163 mode. 3164B for a 3164 terminal in 3164 mode. 4975-01A, for a 4975-01A 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 terminal or equivalent device 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

BG0459

For example, if the device is connected with a 2095 eight-line controller and a 2096 adapter, specify CODTYPE=ASCII.

Specify ASCII for a 3101 terminal or equivalent device in either block mode or character mode.

Specify ASCII 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 terminal in character mode), but the value cannot be increased dynamically.

This operand is not required for the 3101 terminal or equivalent device 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 terminal or equivalent device in block mode; the system sets TOPM to 0 regardless of your specification.

NHIST= This operand is not required for the 3101 terminal or equivalent device 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 terminal or equivalent device 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 terminal or equivalent device 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 terminal or equivalent device 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 terminal or equivalent device 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 through the 1610 or 2091 controllers and the 2092 adapter, code in mirror image. For information on how to code this operand for ASCII terminals, see "ASCII Transmission Codes" on page A-47.

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 through 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, see "ASCII Transmission Codes" on page A-47.

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 terminal or an equivalent device in block mode, specify 9600 if you do not use modems or the maximum speed capacity of the modem you use. A faster bit rate of 19200 is available for 3151, 3161, 3163, or 3164 displays attached to the Feature Programmable Card (#2095/2096) and RPQ D02350).

For the 4975-01A printer attached through a Feature Programmable Card (#2095/2096), specify BITRATE=1200 or 4800.

For ACCA terminals, you may specify BITRATE=0. You should only specify this bit rate if the terminal is attached through the #2095 controller and the #2096 adapter and uses the external clock.

Notes:

1. The MFA (#1310) can run only at 1200, 2400, 4800, and 9600 bits per second.
2. For the Remote Support Link, specify BITRATE=1200. (See "Examples and Defaults" on page A-98 for examples.)

Default BITRATE is 300 bits per second; the default for MFA is 1200 bits per second.

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 terminal or equivalent device in block mode, specify HIGH for best performance.

Default RANGE is HIGH.

LMODE = For ACCA devices, the following options are available:

RS422 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, through the RS-232-C port.

SWITCHED For a connection that is switched. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only. Request-to-Send is not activated permanently.

SWPRTS For a connection that is switched. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only. It permanently activates Request-to-Send (RTS). This is equivalent to installing the RTS jumper on a single-line adapter.

PTTOPT For a connection that is point-to-point nonswitched. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only. Request-to-Send is not activated permanently.

PTTPRTS For a connection that is point-to-point nonswitched. For the Multifunction Attachment Feature #1310, the terminal must be attached on the base address only. It permanently activates Request-to-Send (RTS). This is equivalent to installing the RTS jumper on a single-line adapter.

The LMODE specified should match the jumpers on the controller cards. The Multifunction Attachment does not have jumpers. (See Appendix B, "Customizing Adapters with Hardware Jumpers" for hardware considerations.)

Note: Specify LMODE=SWPRTS for the Remote Support Link. (See "Examples and Defaults" on page A-98 for examples.)

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 or equivalent terminals attached through the Multifunction Attachment (#1310) must specify ADAPTER=MFA.

COD= Additional characters, other than the CR=, ATTN=, and LINEDEL= values, that will end a READTEXT operation. 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. You can enter from one to four characters; any more will be ignored. Code ASCII or mirror image ASCII (EBASC) depending upon device type. For a 4975-0IA printer, specify COD=13.

If you specify PACING=XONXOFF, the system automatically includes X'13' (XOFF) as a COD character. Therefore, you can only enter from one to three characters; any additional characters entered will be ignored.

```
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 through the 1610 or 2091 controllers with the 2092 adapter.)

Default CR is B0 for EBASC and 0D for ASCII.

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 through 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 ends 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.)

The attention key can be redefined by coding this statement as one or two characters (2 or 4 hexadecimal digits).

For terminals attached through the 1610 or 2091 controllers and the 2092 adapter, use mirror image.

For the 3101 terminal or equivalent device 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 the *IBM 3101 Display Terminal Description*, GA18-2033.)

For terminals other than the 3101 or equivalent device, the ASCII default is X'1B' (the ESC key). The EBASC default is X'D8'. If only one character (two hexadecimal digits) is coded, then the PF1 key sequence can be only one character.

PF1=

The one or two character (two or four hexadecimal digit) sequence that you want the system to interpret as the PF1 key. If you code two characters, the system interprets successive values as the PF2 key through the PF8 key. If you code only one character, only PF1 is available.

For terminals attached through a 1610 controller or a 2091 controller with a 2092 adapter, use mirror image.

For the 3101 terminal or equivalent device in either character or block mode, the system defines the PF1 key as the default PF1 key. The character sequence is X'1B61' for ASCII or X'D886' for EBASC. 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 you code only one character (two hexadecimal digits), the system makes the second character binary zeros, in which case only PF1 is available.

For the 3101 display terminal or equivalent device, defaults are listed in the section titled "Examples and Defaults" on page A-98.

For terminals other than the 3101 or equivalent device, the default is binary zeros and no PF function is available. If the ATTN= defaulted to one character, or was coded as only one character, then the PF1 sequence can be only one character.

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 a 3101 terminal or equivalent device, 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 a 3101 terminal or equivalent device 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*.

PART= The partition with which the terminal is normally associated. Code 1–32 for the 4956 models J and K. Code 1–16 for the 4956 models E, 60E, and H (with Extended Address Mode defined) or 1–8 for all others. The default is partition 1.

You can change the partition assignment at execution time with the $CP operator command described in the *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 RS422, the defaults are 1,1,1,1. For PTTOPT and PTTPRTS, the defaults are 2,2,10,2 and for SWITCHED and SWPRTS 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 the terminal as a valid spoolable device.

NO (the default), to specify the terminal is not a valid spoolable device. However, you can redefine this device as a spoolable device at a later time using $TERMUT1.

PACING= XONXOFF to activate XON/XOFF pacing protocol. The default is PACING=NO.

Be sure to include modules IOSACCA and IOS4975A for XON/XOFF support.

XON/XOFF pacing is valid only for the Multifunction Attachment Feature (#1310) and the Feature Programmable Attachment (#2095/2096).

Refer to the "Line Sharing and XON/XOFF Pacing" chapter in the *Communications Guide* for more information on XON/XOFF.

SHARING= YES, to specify this device as the primary device attached to a shared line.

NO to inhibit the use of line sharing with this device.

MODE= must be coded as either 3101B, 3161B, 3163B, or 3164B.

Line sharing is valid only for the Multifunction Attachment Feature (#1310) and the Feature Programmable Attachment (#2095/2096).

Line sharing support allows two distinct line protocols. If the primary device is a 3151, 3161, 3163, or 3164, the XON/XOFF line protocol must be used to communicate with the work station. If the primary device is a PC emulating a 3101B, the IWS direct print protocol must be used to communicate with the work station.

You must select the line protocol by coding the PACING= parameter for the primary device. If the primary device is a 3151 or 3161, code MODE=3161B and PACING=XONXOFF. If the primary device is a 3163 or 3164, code MODE=3163B or MODE=3164B, respectively, and code PACING=XONXOFF. If the primary device is a PC emulating a 3101B, code MODE=3101B, and PACING=NO (the default).

ATTN=NO must not be specified.

You must include the line sharing module, IOSHARE, at system generation.

See the SHARING= parameter of the 4201/4202 TERMINAL statement for information on the attached printer.

END= YES, for the last TERMINAL statement in a system definition module. The default is END=NO.

Examples and Defaults

Default values for optional operands 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 through the 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 terminal or equivalent device. Address assignments are for illustration only. The default values require that you set space parity in the parity bit selection setup switches. (See Appendix C, "Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals" on page C-1 and *IBM 3101 Display Terminal Description*, GA18-2033 for information on switch settings.)

Example 2: IBM 3101 terminal or equivalent device (connected with 1610 single-line controller or 2091 8-line controller with 2092 4-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
```

Example 3: IBM 3101 terminal or equivalent device in block mode (connected with a 1610 single-line controller or 2091 8-line controller with 2092 4-line adapter).

Note: For the 3151, 3161, 3163, and 3164, see the MODE= explanation.

```
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 terminal or equivalent device.

The following two examples contain sample values for defining the 3101 terminal or equivalent device 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, "Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals" on page C-1 for recommended switch settings. Address assignments are for illustration only.

Example 4: IBM 3101 terminal or equivalent device 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
```

Example 5: IBM 3161 terminal in 3161 block mode (connected with a 2095 eight-line controller and a 2096 four-line adapter).

```
label    TERMINAL  DEVICE=ACCA,ADDRESS=61,MODE=3161B,          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 terminal or equivalent device when it is connected through the #1310 adapter. These examples are intended to guide you in coding your TERMINAL statement.

Example 6: IBM 3101 terminal or equivalent device 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 3163 terminal 3163 block mode connected to a #1310 (MFA) that is not the base address.

```
label    TERMINAL  DEVICE=ACCA,ADDRESS=59,MODE=3163B,          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
```

Example 8: The following example contains sample values for defining the 4975-01A printer when it is connected through 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=(13),              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 1: 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 2: 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 3: With an MFA attachment.

```
label    TERMINAL  DEVICE=ACCA,ADDRESS=58,BITRATE=1200,        C
               LMODE=SWPRTS,ADAPTER=MFA,PF1=1B61,              C
               CODTYPE=ASCII,ATTN=1B68,PAGSIZE=24,PART=3,      C
               SCREEN=YES
```

Example 4: 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
```

Asynchronous Line Sharing

The following examples contain statements for connecting ACCA terminals to a shared line. PACING is also active.

Example 1: The following example contains sample values for defining a 3163 terminal when it is connected through a shared line to an RS-422-A port of the Multifunction Attachment Feature (#1310).

```
label      TERMINAL  DEVICE=ACCA,ADDRESS=5B,MODE=3163B,        C
                     BITRATE=9600,PART=4,CODTYPE=ASCII,        C
                     LMODE=RS422,ADAPTER=MFA,SHARING=YES,      C
                     PACING=XONXOFF,END=NO
```

Example 2: The following example contains sample values for defining a Personal Computer emulating a 3101B terminal connected through a shared line to the Feature Programmable Attachment (#2095/2096).

```
label      TERMINAL  DEVICE=ACCA,ADDRESS=AB,MODE=3101B,        C
                     BITRATE=9600,PART=4,CODTYPE=ASCII,        C
                     LMODE=LOCAL,ADAPTER=EIGHT,SHARING=YES,    C
                     PACING=NO,END=NO
```

TTY-Type Terminals

A TERMINAL definition statement with DEVICE=TTY locally attaches teletypewriter terminals and IBM 3101 terminals or equivalent devices 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 alphanumeric 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 terminal or equivalent device in character mode or another ASCII terminal attached through the Teletypewriter Adapter (#7850).
ADDRESS=	The address (in hexadecimal) of the device.
MODE=	3101C, for a 3101 terminal or equivalent device 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.

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. See "ASCII Transmission Codes" on page A-47.

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, see "ASCII Transmission Codes" on page A-47.

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= operand 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, see "ASCII Transmission Codes" on page A-47.

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 ends 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, see "ASCII Transmission Codes" on page A-47.

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 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.

(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 terminal or equivalent device, defaults are listed in "Examples and Defaults" on page A-107.

For the 3101 terminal or equivalent device 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 a 3101 terminal or equivalent device. When the screen fills up, you do ***not*** have to press enter for the output to continue.

PART= The partition with which the terminal is normally associated. Code 1–32 for the 4956 models J and K. Code 1–16 for the 4956 models E, 60E, and H (with extended address mode defined) or 1–8 for all others. 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 the 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.

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 through the 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 terminal or equivalent device 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, "Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals" on page C-1 for information on switch settings.)

Example 2: IBM 3101 terminal or equivalent device 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 terminal or equivalent device.

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 operands in accordance with the hardware jumpers, matching the setting in the other processor. Also, the LINSIZE operand 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 (refer to *IBM Series/1 Asynchronous Communications Feature Description*, GA34-0243) 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.

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' (refer to *IBM Series/1 Asynchronous Communications Feature Description*, GA34-0243) 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.

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 ends 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
```

Interprogram Communication - Virtual Terminals

A TERMINAL definition statement with DEVICE=VIRT defines interprogram communication or $VIRLOG support. 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 or $VIRLOG support.
ADDRESS=	The label of the other virtual terminal for interprogram communication.
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.

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 an operand is not shown, then it is not relevant for the device. Address assignments are for illustration only.

Example 1: Terminal statements for the Remote Management Utility using the PASSTHRU function.

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
```

Example 3: Defining terminals for $VIRLOG support in the $EDXDEF data set.

```
$$SYSLOG   TERMINAL  DEVICE=VIRT,ADDRESS=X$SYSLOG
X$SYSLOG   TERMINAL  DEVICE=VIRT,ADDRESS=$$SYSLOG,SYNC=YES
```

Note: See also "SYSMSG - Define System Message Destination" on page A-21 for additional information on $VIRLOG support.

General Purpose Interface Bus

A TERMINAL definition statement with DEVICE=GPIB defines a General Purpose Interface Bus (GPIB). Refer to the *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 terminal, the default is ASCII X'1B' (the ESC key). The mirror image of X'1B' is X'D8'. **Note:** If the terminal being defined is specified in the HDCOPY= operand 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 with which the terminal is normally associated. Code 1 – 32 for the 4956 models J and K. Code 1 – 16 for the 4956 models E, 60E, and H (with Extended Address Mode defined) or 1 – 8 for all others. 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 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 the *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.

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, you might want to refer to the appropriate hardware manual for your ACCA adapter:

- *IBM Series/1 Synchronous Data Link Control Communications Feature Description*, GA34-0245.
- *IBM Series/1 Asynchronous Communications Feature Description*, GA34-0243 for the 2091/2092 and 1610 attachments.
- *IBM Series/1 Asynchronous Direct 8-Line RS-422-A Adapter Custom Feature*, GA34-1578 for the 2095/2096 attachment (RPQ D02359).

If your terminal configuration includes the IBM 3101 terminal (or its equivalent), see both Appendix C, "Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals" and the *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 A-1.

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 terminal and equivalent devices, see Appendix C, "Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals."

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 appropriate hardware manual.

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 appropriate hardware manual.

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)	Connected 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 can use the #2056 cable.

Terminals Connected Using Digital I/O #1560

Terminal support is provided for digital I/O devices such as the Tektronix[1] 4010 series of terminals equipped with the General Purpose Parallel Interface (Tektronix Custom Feature Number CM021-0109-03, with cable: CM012-0541-00) or terminals naving 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.

[1] Trademark of Tektronix, Inc.

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) HANDSHAKE CONTROL DATA	Set all to LOW
THRESHOLD	+2 volts
MASTER OPTION	None

Driver Card	
LOGIC SENSE (4) STATUS HANDSHAKE INTERRUPT DATA	Set as shown HIGH HIGH LOW 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. When you use GIN mode, you must press the appropriate key followed by the carriage return (CR). You can 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 go to the home address automatically and continue printing without erasing the screen.

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 (or equivalent) terminals
- 4975 printers
- A binary synchronous communications line.

The use of the RS-232-C 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-C or RS-422-A interface mode. Ports 1, 2, and 3 are limited to operations through the RS-422-A interface.

You can substitute the #1310 adapter for other asynchronous adapters subject to the limitations on the interface requirements and supported terminal types stated above.

Appendix C. Configuring and Connecting 3101, 3151, 3161, 3163, and 3164 Display Terminals

The IBM 3101 terminal is supported under the Event Driven Executive in both character and block mode. The 3151 (Models 31 and 41 only), 3161, 3163, and 3164 display terminals are supported in 3101 emulation mode (character, block, and echo modes) or in their own modes (block mode only). This appendix presents information to aid you in planning the link between the Series/1 and these display terminals.

Connecting Your 3101, 3151, 3161, 3163, or 3164 to the Series/1

The 3101, 3151, 3161, 3163, and 3164 terminals are connected to the Series/1 by the attachment features supported by the Event Driven Executive. For each attachment and type of interface, you must set the hardware switches on the 3101 and the software switches on the 3151, 3161, 3163, and 3164. You must also have the attachment card physically jumpered, connect the cables, and specify the appropriate TERMINAL statement (if applicable).

The following chart shows the modes supported by the various attachment features for the 3101 and the 3151, 3161, 3163 and 3164 terminals operating in 3101 emulation mode:

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 C-1. 3101, 3151, 3161, 3163, and 3164 Attachment Features

For each of these attachment features, there are many configurations available. Refer to "Configuring Your 3101, 3151, 3161, 3163, or 3164" on page C-33 for sample configurations for each attachment card option, the cable connections, the switch settings, and the TERMINAL statements for each interface/attachment.

Hardware Setup Switches for the 3101

Figure C-2 on page C-3 shows the hardware switch layout for a 3101. You can select the functions you need by setting these hardware switches. An explanation of the numbered items follows the figure.

Note: See "Software Setup Switches for the 3151, 3161, 3163, and 3164" on page C-7 for an explanation of how the 3151/3161/3163/3164 software switch setups correspond to the 3101 hardware switch setups.

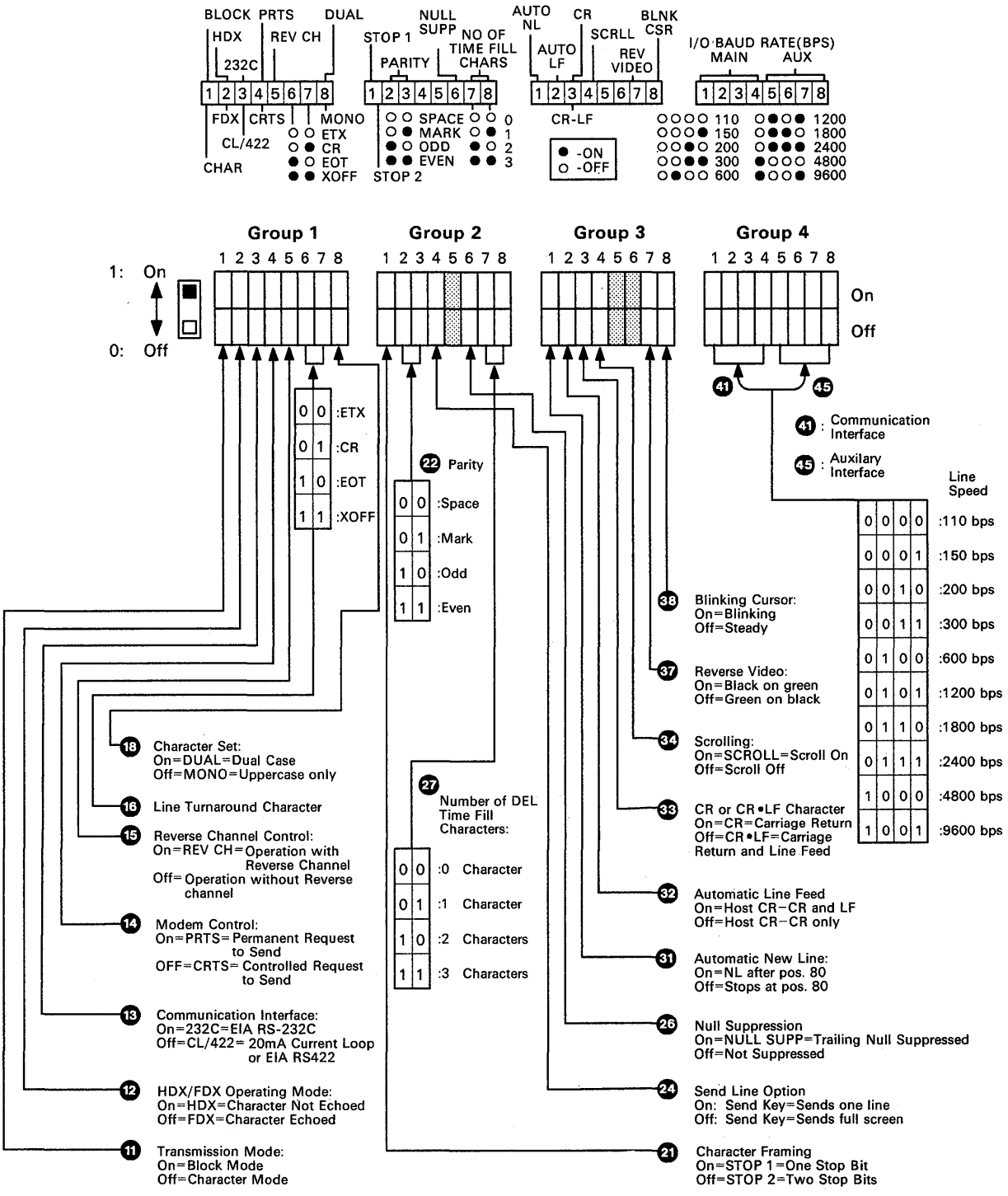


Figure C-2. 3101 Setup Switch Layout

11

In the BLOCK position, the system stores the characters you enter from the keyboard in a buffer. In the CHAR (character) position, the system transmits immediately, character by character, the characters you enter from the keyboard.

12

The FDX position enables full-duplex operation (CHAR mode only). The HDX position enables half-duplex operation. BLOCK mode supports only HDX.

13

The RS-232-C position selects the EIA RS-232-C interface connector. The CL/RS-422-A position selects the 20-mA current loop or EIA RS-422-A interface connector.

14

In the PRTS position, the Request-to-Send signal stays on. In the CRTS position, the Request-to-Send signal is controlled by the 3101.

15

This switch is used with EIA RS-232-C only. The ON position activates a half-duplex communication facility with a reverse channel. The OFF position is for half-duplex communication facility without a reverse channel.

16

This switch is for use with character mode. You select the turnaround character in one of the following ways:

- For CR (01), press the ALT key and the letter M at the same time.
- For EOT (10), press the ALT key and the letter D at the same time.
- For ETX (00), press the ALT key and the letter C at the same time.
- For XOFF (11), press the ALT key and the letter S at the same time.

18

The DUAL position allows you to use both upper- and lower-case letters. The MONO position allows you to use only the upper-case letters. (The 3161, 3163, and 3164 support DUAL mode only.)

21

The STOP 1 position selects one stop bit for each frame. The STOP 2 position selects two stop bits for each frame. The setting depends on the line speed. 110 bps generally uses two stop bits while 150 bps and above use one stop bit.

22

The ODD or EVEN parity position generates the proper bit setting for the data being transmitted and checks for the parity of all received data. The MARK position forces the parity bit to 1 for each character transmitted, while the SPACE position forces the parity bit to 0. The 3101 does not check parity on received data for MARK or SPACE.

24

The ON position reverses the function of the SEND LINE key and the SEND key (an advantage for most time-sharing systems).

26

The ON position suppresses trailing nulls when you perform block transmission or a buffer-print operation. ("Trailing nulls" are null characters that follow the last significant character (EOL or EOF) or the cursor in a line or screen.) The OFF position converts null characters to space characters and transmits them.

27

The setting for these two switches determines the number of time-fill characters (DELs) used for time fill for the print data stream. The choices are:

- For 0 characters, set the switches to 0 and 0 (OFF, OFF).
- For 1 character, set the switches to 0 and 1 (OFF, ON).
- For 2 characters, set the switches to 1 and 0 (ON, OFF).
- For 3 characters, set the switches to 1 and 1 (ON, ON).

31

The ON position automatically moves the cursor to the first position of the new line when you type the 80th character. The OFF position causes an audible alarm after the 72nd character, and after the 80th character, the cursor will not move.

32

The ON position causes the cursor to move automatically to the first position of the next line after a carriage return from the host. The OFF position causes the cursor to move to the first position of the ***same*** line after a carriage return from the host.

33

The ON position causes the NEW LINE key to generate the CR character. The OFF position causes the NEW LINE key to generate the CR/LF characters.

34

The ON position causes the existing 24th line to disappear and a new, clean 24th line to appear. The OFF position causes the cursor to stop at the 80th column of line 24. If you press the Cursor Home key, the new line begins at the top of the screen, replacing the existing line character by character.

37

The ON position displays dark characters on a green screen. The OFF position displays green characters on a dark screen.

38

The ON position provides a blinking cursor.

41

Depending on your line speed requirements, set the four switches as follows:

- For 110 bps, 0 0 0 0 (OFF, OFF, OFF, OFF)
- For 150 bps, 0 0 0 1 (OFF, OFF, OFF, ON)
- For 200 bps, 0 0 1 0 (OFF, OFF, ON, OFF)
- For 300 bps, 0 0 1 1 (OFF, OFF, ON, ON)
- For 600 bps, 0 1 0 0 (OFF, ON, OFF, OFF)
- For 1200 bps, 0 1 0 1 (OFF, ON, OFF, ON)
- For 1800 bps, 0 1 1 0 (OFF, ON, ON, OFF)
- For 2400 bps, 0 1 1 1 (OFF, ON, ON, ON)
- For 4800 bps, 1 0 0 0 (ON, OFF, OFF, OFF)
- For 9600 bps, 1 0 0 1 (ON, OFF, OFF, ON).

45

The auxiliary interface works at the same line speed as the communication interface for the monitor mode (CHAR or BLOCK). If you are not using monitor mode, set these switches according to the print capability of the attached printer and /or buffer.

Software Setup Switches for the 3151, 3161, 3163, and 3164

The 3151, 3161, 3163, and 3164 do not have hardware switches. Instead, you set the configuration you need by selecting options on a menu. See "Setting Switches for Terminals in 3101 Emulation Mode Using a 3151 Display Station" if you are using a 3151 Display Station. See "Setting Switches for Terminals in 3101 Emulation Mode Using a 3161, 3163, or 3164" on page C-11 for information on setting up 3101 emulation mode for 3161, 3163, and 3164 terminals. See "Setting Switches for 3161, 3163, or 3164 Terminals in Their Own Block Mode" on page C-16 for information on setting up 3161/3163/3164 in their own block mode.

Setting Switches for Terminals in 3101 Emulation Mode Using a 3151 Display Station

For the 3151 terminals, press and hold the control key and then press the setup key so that you can select your software terminal switches that are displayed on subsequent menus.

Following the above action, the first "General" menu appears as in the following example:

```
                    S E T U P   M E N U
GENERAL          COMMUNICATION       KEYBOARD/PRINTER          FUNCTION

Machine Mode          IBM 3151       Forcing Insert                 OFF
Screen                  NORMAL       Tab                          FIELD
Row and Column        24 x 80
Scroll                   JUMP
Auto LF                    ON
CRT Saver                 OFF       Term.ID_____________
Line Wrap                  ON
```

Note: When the terminal is installed initially, the values displayed are default values. Subsequent software setup switch sessions can display the current state (or saved) setup switch status.

On the displayed menus, functions that are applicable to the terminal itself are not a concern in terminal session operations (for example, CRT saver, where the screen blanks during periods of inactivity to protect the tube from burnout).

To make your terminal agree with its specified TERMINAL statement and adapter card hookup, you move the cursor to the first field on the menu that requires modification. Then you press the space bar on the keyboard to advance the value within the field selected to the value required. You continue on by moving the cursor to the next field requiring alteration. Again, use the space bar to advance the value displayed to the value you desire.

When you have completed the changes you need to make on the General menu, press the SEND key. The Communication menu appears as in the following example:

```
                    S E T U P   M E N U
GENERAL          COMMUNICATION       KEYBOARD/PRINTER          FUNCTION

Operating Mode          BLOCK
Line Speed (bps)         9600       Line Control                  PRTS
Word Length (bits)          7       Break Signal (ms)              500
Parity                    ODD       Send Null Suppress              ON
Stop Bit                    1       Pacing                         OFF
Turnaround Character      ETX
```

To change a setting on the select menu, move the cursor to the selection you want to change. Then press the space bar until the option you want appears on the screen.

When you have completed your selections, press the SEND key.

SEND key action causes the "Keyboard/Printer" menu to appear as in the following example.

```
                    S E T U P   M E N U
GENERAL          COMMUNICATION       KEYBOARD/PRINTER          FUNCTION

KEYBOARD                            PRINTER
 Enter                  RETURN       Line Speed (pbs)              9600
 Return                 FIELD        Word Length (bits)               7
 New Line                 CR         Parity                         ODD
 Send                   PAGE         Stop Bit                         1
 Insert Character       MODE         Characters                NATIONAL
```

This menu supplies values for proper keyboard usage, as well as values that are required if you have a printer connected to your terminal. The left column relates to keyboard activity; the right column relates to the printer.

To change a setting on the select menu, move the cursor to the selection you want to change. Then press the space bar until the option you want appears on the screen. When you have completed the changes you need to make, press the SEND key. The final menu, the "Function" menu appears, as in the following example:

```
                    S E T U P   M E N U
GENERAL          COMMUNICATION       KEYBOARD/PRINTER          FUNCTION

   Recall                    Save                      Default

 Reset Terminal              Clear Status              Mode Adjust
```

You use the above function to activate the terminal to the values you have selected previously. This is done by moving the cursor to the SAVE field. Press the space bar (not the SEND key) to activate the terminal to the values you have selected. When the value is activated, the system displays a "completed" message.

For additional information on the 3151 terminal and options for the setup menus, refer to the *IBM 3151 ASCII Display Station: Guide to Operations*, GA18-2633, and/or the *IBM 3151 ASCII Display Station: Reference Manual*, GA18-2634.

The following illustrates typical 3151 menu setups for a 3151 terminal connected to a multifunction adapter (port 1) for 3101 emulation in block mode, character mode, and 3151 native mode (block) all using the RS-422-A interface.

Menus	**3101 Emulation Block Mode**	**3101 Emulation Character Mode**	**3151 Native Mode**
General			
Machine Mode	IBM 3101	IBM 3101	IBM 3151
Screen	Normal	Normal	Normal
Row & Column	24X80	24X80	24X80
Scroll	Jump	Jump	Jump
Auto LF	ON	OFF	ON
CRT Saver	OFF	OFF	OFF
Line Wrap	ON	ON	ON
Forcing Insert	OFF	OFF	OFF
Tab	FIELD	FIELD	FIELD
Terminal ID			
Communication			
Operating Mode	BLOCK	CHARACTER	BLOCK
Line Speed	9600	9600	9600
Word Length	7	7	7
Parity	SPACE	SPACE	SPACE
Stop Bit	2	2	2
Turnaround Character	CR	CR	CR
Line Protocol	RS-422-A	RS-422-A	RS-422-A
Line Control			
Break Signal (ms)	500	500	500
Send Null Suppress	ON	ON	ON
Pacing			
Keyboard			

Figure C-3 (Part 1 of 2). Typical 3151 Menu Setups

Menus	3101 Emulation Block Mode	3101 Emulation Character Mode	3151 Native Mode
Enter	Return	Return	Return
Return	Field	Field	Field
New Line	CR	CR	CR
Send	Line	Line	Line
Insert Character	Mode	Mode	Mode

Figure C-3 (Part 2 of 2). Typical 3151 Menu Setups

Notes:

1. For other configuration possibilities, consult the *IBM 3152 ASCII Display Station Guide to Operations.*
2. The 3151 Model II does not support 3101 emulation mode; only Models 31 and 41 support 3101 emulation.
3. The 3151 operating in its own mode only supports block mode.

Setting Switches for Terminals in 3101 Emulation Mode Using a 3161, 3163, or 3164

To display the main "setup menu" for a 3161, 3163, or 3164 in 3101 emulation mode, press and hold the control key, then press the select key. The main screen appears as in the following example:

```
                    Setup Menu

Machine Mode              IBM 3101

Operating Mode            BLOCK


  Interface                   RS-232-C
  Line Control                PRTS
  Line Speed (bps)            9600
  Parity                      SPACE
  Turnaround Character        CR
  Stop Bit                    2



  When all finished, press send.  Press Setup to quit.

LOCAL
```

To change a setting on the setup menu, move the cursor to the selection you want to change. Then press the space bar until the option you want appears on the screen. Figure C-4 on page C-12 shows the setup menu options and how they correspond to the 3101 switch options.

Note: Some of the options for 3101 emulation mode appear on the second menu for the 3161/3163/3164, illustrated in Figure C-5 on page C-14.

Use the following figure to help you convert 3101 setup switch settings to 3161, 3163, and 3164 setup menu parameters for 3101 emulation mode.

Comparing Setup Menu Options

3101	3101 Emulation Mode
None	Machine Mode IBM 3101
Group 1:	
Switches 1 and 2	Operating Mode
ON ON	BLOCK (Block and HDX)
OFF ON	CHARACTER
OFF OFF	ECHO (for TTY)
Switch 3	Interface
ON	RS-232-C
OFF	RS-422-A
Switch 4	Line Control
ON	PRTS
OFF	CRTS
	IPRTS (not supported)
Switch 5	Reverse Channel Control
ON	None
OFF	None
Switches 6 and 7	Line Turnaround Character
OFF OFF	ETX
OFF ON	CR
ON OFF	EOT
ON ON	
	DC3
Switch 8	Character Set
ON (dual case)	Controlled by
OFF (uppercase)	CAPS key
Group 2:	
Switch 1	Stop Bit
ON	1
OFF	2

Figure C-4 (Part 1 of 3). Comparing Setup Menu Options for the 3101 and 3101 Emulation Mode

Comparing Setup Menu Options (continued)

3101				3101 Emulation Mode
Switches 2 and 3				Parity
OFF	OFF			SPACE
OFF	ON			MARK
ON	OFF			ODD
ON	ON			EVEN
				NO
Switch 4				Send Line
See Figure C-6 on page C-15				
Switch 6				Null Suppression
See Figure C-6 on page C-15				
Switches 7 and 8				# of Time Fill Characters
OFF	OFF	(0)		
OFF	ON	(1)		Always
ON	OFF	(2)		0
ON	ON	(3)		
Group 3: See Figure C-6 on page C-15				
Group 4:				
Switches 1, 2, 3, 4				Line Speed
Main Port Baud Rate				
				50 bps
				75 bps
OFF	OFF	OFF	OFF	110 bps
				134.5 bps
OFF	OFF	OFF	ON	150 bps
OFF	OFF	ON	OFF	200 bps
OFF	OFF	ON	ON	300 bps
OFF	ON	OFF	OFF	600 bps
OFF	ON	OFF	ON	1200 bps
OFF	ON	ON	OFF	1800 bps
OFF	ON	ON	ON	2400 bps
				3600 bps
ON	OFF	OFF	OFF	4800 bps
				7200 bps
ON	OFF	OFF	ON	9600 bps
				19200 bps

Figure C-4 (Part 2 of 3). Comparing Setup Menu Options for the 3101 and 3101 Emulation Mode

Comparing Setup Menu Options (continued)

3101				3101 Emulation Mode
Switches 5, 6, 7, 8				Line Speed
Auxiliary Port Baud Rate				
OFF	OFF	OFF	OFF	110 bps
OFF	OFF	OFF	ON	150 bps
OFF	OFF	ON	OFF	200 bps
OFF	OFF	ON	ON	300 bps
OFF	ON	OFF	OFF	600 bps
OFF	ON	OFF	ON	1200 bps
OFF	ON	ON	OFF	1800 bps
OFF	ON	ON	ON	2400 bps
ON	OFF	OFF	OFF	4800 bps
ON	OFF	OFF	ON	9600 bps

Figure C-4 (Part 3 of 3). Comparing Setup Menu Options for the 3101 and 3101 Emulation Mode

The remaining options you can change for 3101 emulation mode are on the "select menu." To get the select menu to appear at the bottom of your screen, press the select key. The select screen appears as in the following example. A list of the setup options follows the example.

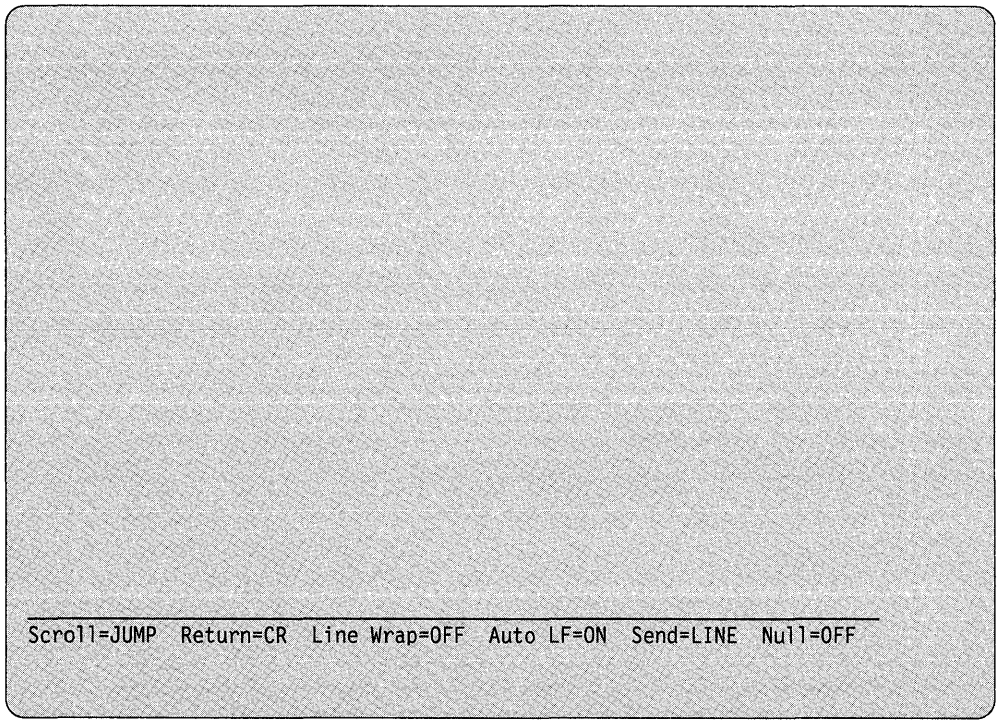


Figure C-5. Select Menu for 3101 and 3161/3163/3164

To change a setting on the select menu, move the cursor to the selection you want to change. Then press the space bar until the option you want appears on the screen. Figure C-6 shows the select menu options and how they compare to the 3101 switch options.

Comparing Select Menu Options

3101	3101 Emulation Mode
Group 2	
Switch 4	Send=
ON	Line
OFF	Page
Switch 6	Null Supp=
ON	ON
OFF	OFF
Group 3	
Switch 1	Line Wrap=
ON	ON
OFF	OFF
Switch 2	Auto LF=
ON	ON
OFF	OFF
Switch 3	Return=
ON	CR
OFF	CR-LF
Switch 4	Scroll=
ON	JUMP (for 3163 only)
ON	ON (for 3161 only)
OFF	OFF
Switch 7	Reverse Video
ON	None
OFF	
Switch 8	Blinking Cursor
ON	Controlled by
OFF	ALT/CSR key

Figure C-6. Comparing Select Menu Options for 3101 and 3101 Emulation Mode

Setting Switches for 3161, 3163, or 3164 Terminals in Their Own Block Mode

To display the main "setup menu" for the 3161, 3163, or 3164 in their own block mode, press and hold the control key, then press the select key. The main screen appears for the 3164 as in the following example:

Note: The "Machine Mode" option differs according to your terminal model.

```
                    Setup Menu

Machine Mode              IBM 3164

Operating Mode            BLOCK

                          Main Port      Auxiliary Port

  Interface                  RS-232-C
  Line Control               PRTS
  Line Speed (bps)           19200              19200
  Parity                     SPACE              ODD
  Turnaround Character       CR
  Stop Bit                   2                   1
  Word Length (bits)         7                   7
  Response Delay (ms)        100
  Break Signal (ms)          500

Terminal ID _____________ up to 20 alphanumeric
                          characters are allowed

  When all finished, press send.  Press Setup to quit.

LOCAL
```

To change a setting on the setup menu, move the cursor to the selection you want to change. Then press the space bar until the option you want appears on the screen. Figure C-7 on page C-17 shows the setup menu options and what you can specify for each option.

Note: Other options are available on the "select menu" for the 3161/3163/3164, illustrated in Figure C-8 on page C-19.

Available Setup Menu Options for
3161/3163/3164 Block Mode

Machine Mode
- IBM 3161
- IBM 3163
- IBM 3164

Operating Mode
- BLOCK (Block and HDX)

	Main Port	Auxiliary Port
Interface	RS-422-A RS-232-C	RS-232-C
Line Control	CRTS PRTS IPRTS	
Line Speed	50 bps 75 bps 110 bps 134.5 bps 150 bps 200 bps 300 bps 600 bps 1200 bps 1800 bps 2400 bps 3600 bps 4800 bps 7200 bps 9600 bps 19200 bps 38400 bps	15 bps 75 bps 110 bps 134.5 bps 150 bps 200 bps 300 bps 600 bps 1200 bps 1800 bps 2400 bps 3600 bps 4800 bps 7200 bps 9600 bps 19200 bps

Figure C-7 (Part 1 of 2). Setup Menu Options for the 3161/3163/3164 (Block Mode)

Available Setup Menu Options for
3161/3163/3164 Block Mode (continued)

	Main Port	Auxiliary Port
Parity	SPACE	SPACE
	MARK	MARK
	ODD	ODD
	EVEN	EVEN
	NO	NO
Turnaround Character	CR	
	EOT	
	DC3	
	ETX	
Stop Bit	1	1
	2	2
Word Length (bits)	7	7
	8	8
Response Delay (ms)	100	
	0	
Break Signal(ms)	170	
	500	

Figure C-7 (Part 2 of 2). Setup Menu Options for the 3161/3163/3164 (Block Mode)

Refer to the hardware documentation for information on the recommended software switch settings.

The remaining options you can change are on the "select" menus. To get the first of three select menus to appear at the bottom of your screen, press the select key. The select screen appears as in the following example. To get to subsequent screens, press the send key.

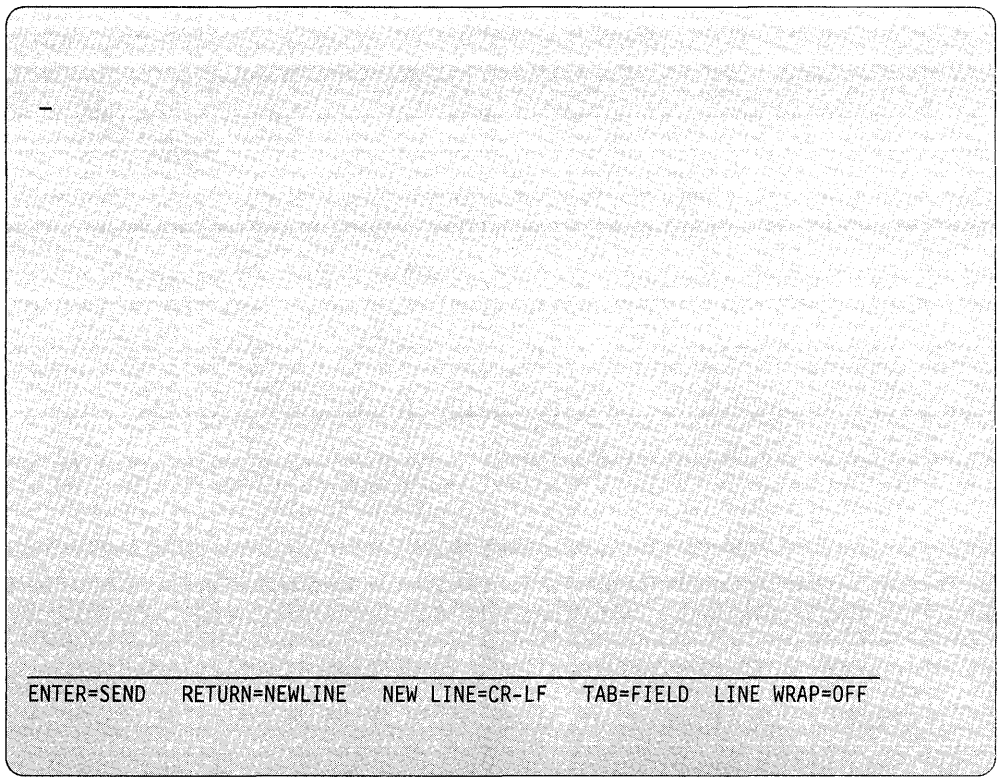


Figure C-8. Select Menu for 3161/3163/3164 in Their Own Block Mode - Screen 1

To change a setting on the select menu, move the cursor to the selection you want to change. Then press the space bar until the option you want appears on the screen. Figure C-11 on page C-22 shows the select menu options and what you can specify for each option.

To get to screen 2, press the Send key.

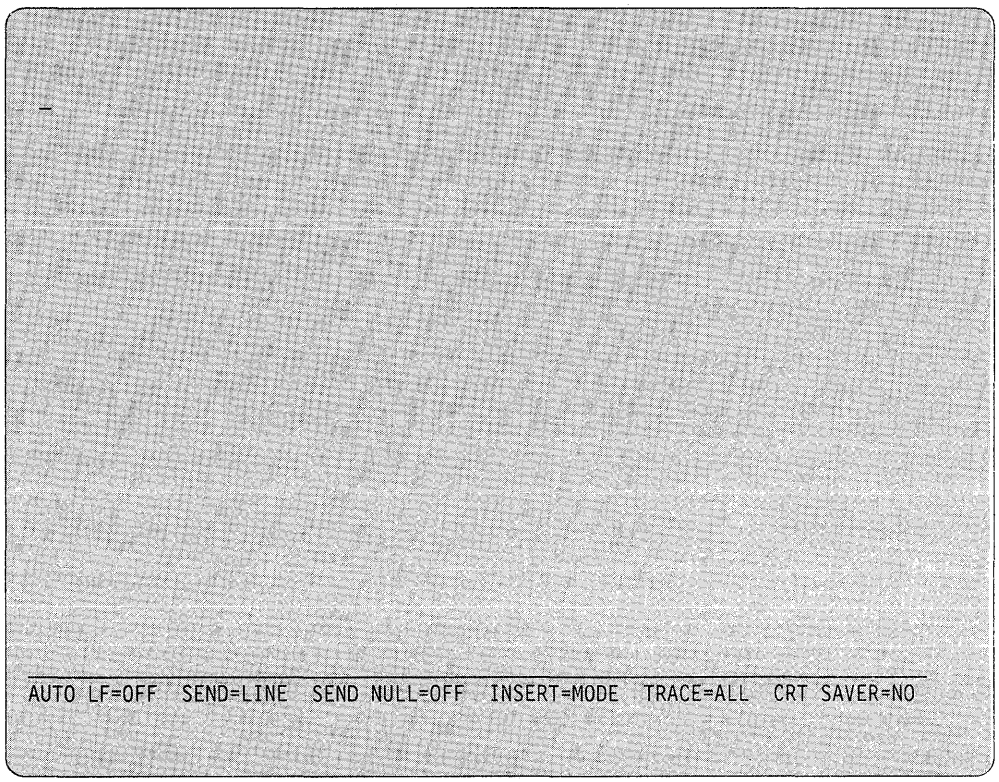


Figure C-9. Select Menu for 3161/3163/3164 in Their Own Block Mode - Screen 2

To change a setting on the select menu, move the cursor to the selection you want to change. Then press the space bar until the option you want appears on the screen. Figure C-11 on page C-22 shows the select menu options and what you can specify for each option.

To get to screen 3, press the Send key.

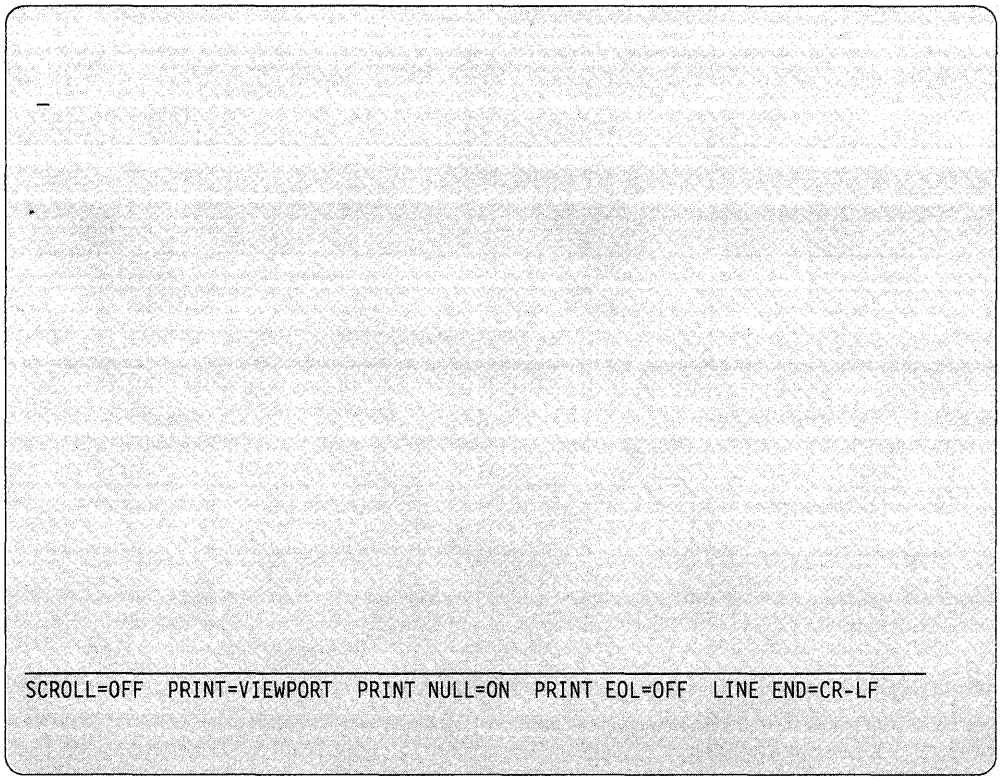


Figure C-10. Select Menu for 3161/3163/3164 in Their Own Block Mode - Screen 3

To change a setting on the select menu, move the cursor to the selection you want to change. Then press the space bar until the option you want appears on the screen. Figure C-11 on page C-22 shows the select menu options and what you can specify for each option.

Available Select Menu Options

Option	3161/3163/3164
ENTER	SEND RETURN
RETURN	FIELD NEW LINE
NEW LINE	CR CR-LF
TAB	FIELD COLUMN
LINE WRAP	OFF ON
AUTO LF	OFF ON
SEND	LINE PAGE
SEND NULL	OFF ON
INSERT	MODE SPACE
TRACE	ALL RECEIVE SEND
CRT SAVER	NO 5 10 15

Figure C-11 (Part 1 of 2). Available Select Menu Options for the 3161/3163/3164 (Block Mode)

Available Select Menu Options	
Option	3161/3163/3164
SCROLL	OFF ON (3161 only) SMOOTH (3163/64 only) JUMP (3163/64 only)
PRINT	VIEWPORT SCREEN PAGE (3163/64 only)
PRINT NULL	ON OFF
PRINT EOL	ON OFF
LINE END	CR-LF CR

Figure C-11 (Part 2 of 2). Available Select Menu Options for the 3161/3163/3164 (Block Mode)

Refer to the hardware documentation for information on the recommended software switch settings.

Attaching 3101, 3151, 3161, 3163, and 3164 Display Terminals

The following sections show you how to set your terminal switches according to the attachment you are using.

3101 and 3101 Emulation Display Terminals in Character Mode

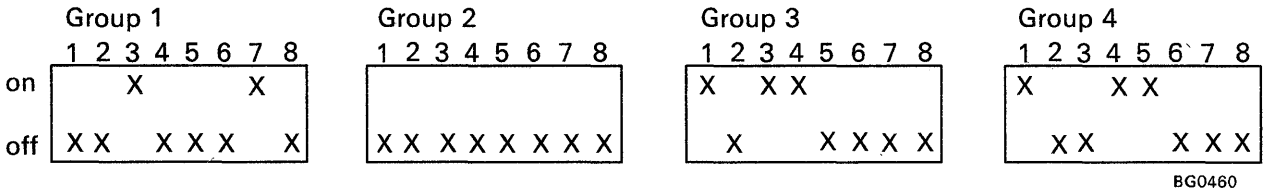
You can connect a 3101 or 3151, 3161, 3163, or 3164 (in 3101 emulation mode only) in character mode to the Series/1 through five attachment features:

- #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 B-1. Each section presents the 3101 hardware switch settings. For a 3151 3161, 3163, or 3164 in 3101 emulation mode, see Figure C-4 on page C-12 and Figure C-6 on page C-15.

Attachment through the #7850 Teletypewriter Adapter

3101 Hardware Switch Settings: For attachment through the #7850 adapter with an EIA interface, you can set the 3101 hardware setup switches as follows:



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 hardware 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 C-16 on page C-34.

Cable connections: Figure C-12 shows the 3101, 3151, 3161, 3163, or 3164 character mode) attachment with the #7850 teletypewriter adapter.

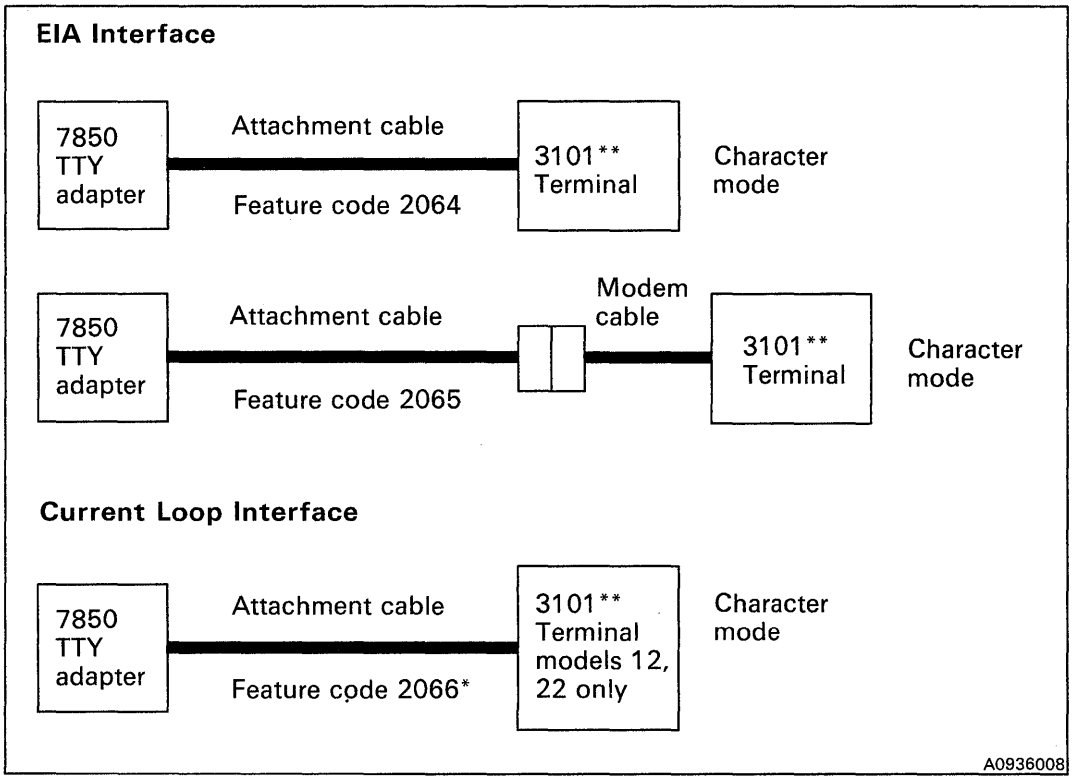


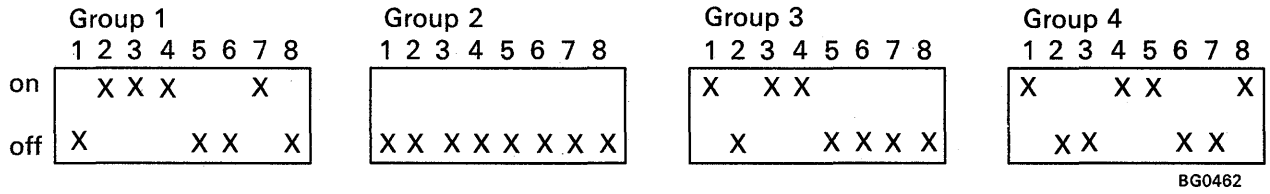
Figure C-12. 3101, 3151, 3161, 3163, and 3164 Terminal with #7850 Adapter

Note: *This cable is unmodified or modified according to the source of the transmit current. See Figure C-16 on page C-34.

**Or supported terminal emulating a 3101 terminal for attachment cables relating to terminals other than 3101. See related terminal associated hardware publication.

Attachment through the #1310, #1610, #2091/#2092, or #2095/#2096

3101 Switch Settings: For attachment through the #1610 controller, the #2091 controller with #2092 adapter, or the #2095 controller with #2096 adapter, you can set the 3101 setup switches as follows:



These switch settings and "Attachment options" on page C-27 correspond to the configurations numbered 4, 5, and 6 in Figure C-16 on page C-34.

The switch settings for the #1310 are the same except that switch 3 of group 1 must be in the "off" (RS-422-A) position if the 3101 terminal is not connected to Port 0. A 3101 attached to Port 0 of a #1310 can operate through either the RS-232-C or RS-422-A interface. Only the 3101 model 23 supports both block mode and the RS-422-A interface. Model 13 supports the RS-422-A 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.

Attachment options: The jumpers for the #1610 controller, the #2091 controller with #2092 adapter, and the #2095 controller with #2096 should have Carrier Detect (CD), Data Terminal Ready (DTR), and Request to Send (RTS) jumpered on. Also, the HIGH- or LOW-speed option must be jumpered to reflect the speed set in the 3101, 3151, 3161, 3163, and 3164 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, 3151, 3161, 3163, and 3164 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-232-C (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, 3151, 3161, 3163, or 3164 setup switches.

Cable connections: See Figure C-14 on page C-30 when your 3101, 3151, 3161, 3163, or 3164 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 C-13.

Operator function	Key on 3101 in character mode	Characters received in the Series/1 with Space Parity set in the 3101 setup switches		
		Device=ACCA		Device=TTY
		#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'

BG0463

Figure C-13. 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, 3151, 3161, 3163, or 3164 is attached through the #1310 adapter, the #1610 controller, the #2091 controller with #2092 adapter, or the #2095 controller with #2096 adapter.

Special Considerations when Using Half-Duplex Modems with a 3101C, 3151, 3161, 3163, or 3164

While the majority of the 3101, 3151, 3161, 3163, and 3164 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, 3151, 3161, 3163, and 3164 to adhere to the line turnaround protocol observed by half-duplex modems.

3101 and 3151, 3161, 3163, 3164 Terminals in Block Mode

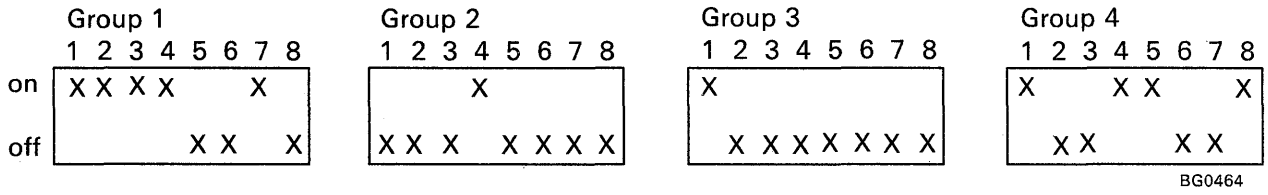
You can connect the IBM 3101, 3151, 3161, 3163, and 3164 display terminal (in block mode) to the Series/1 through four attachments:

- the #1310 adapter
- the #1610 controller
- the #2091 controller with #2092 adapter
- the #2095 controller with #2096 adapter.

Each of the following sections presents the 3101 hardware switch settings. For a 3151, 3161 3163, or 3164 in 3101 emulation mode, see Figure C-4 on page C-12 and Figure C-6 on page C-15; for their own block mode see Figure C-7 on page C-17 and Figure C-11 on page C-22.

3101 Switch Settings

For attachment of the 3101 through these attachment features, you can set the 3101 setup switches as follows:



These switch settings and "Attachment Options" on page C-29 correspond to the configurations numbered 23, 25, and 26 in Figure C-17 on page C-38.

The switch settings for the #1310 are the same except that switch 3 of group 1 must be in the "off" (RS-422-A) position if the 3101 3151, 3161, 3163, or 3164 terminal is not connected to Port 0. A 3101 model 13 or 23 or a 3161 or 3163 attached to Port 0 of a #1310 can operate through either the RS-232-C or RS-422-A interface. Only the 3101 model 23, the 3151, 3161, 3163, and the 3164 support both block mode and the RS-422-A 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 (CD), Data Terminal Ready (DTR), and Request to Send (RTS) 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, 3151, 3161, 3163, or 3164 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 the 3101, 3151, 3161, 3163, or 3164 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, 3151 3161, 3163, and 3164 setup switches.

Cable Connections

Figure C-14 shows the 3101, 3151 3161, 3163, and 3164 (character or block mode) attachment with the #1610 controller or the #2091 controller and the #2092 adapter. Figure C-15 on page C-31 shows the 3101, 3151, 3161, 3163, and 3164 (character or block mode) attachment with the #1310, #2095 controller with the #2096 adapter (EIA interface), and the #2095 with RPQ D02350.

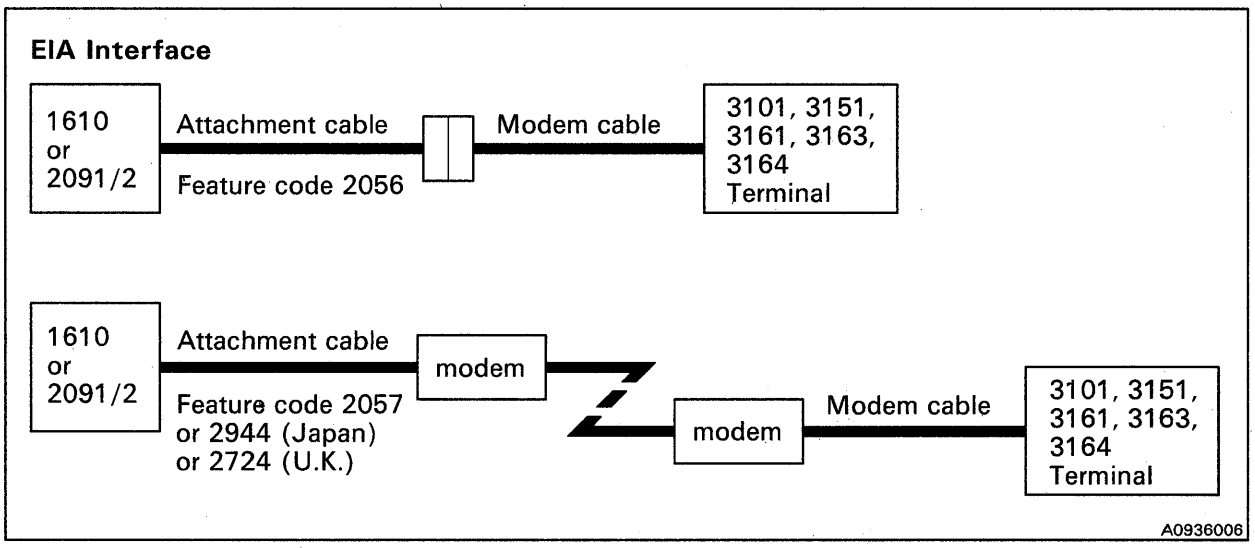


Figure C-14. 3101, 3151, 3161, 3163, 3164 Terminal with #1610 or #2091/#2092

(See Figure C-16 on page C-34 and Figure C-17 on page C-38)

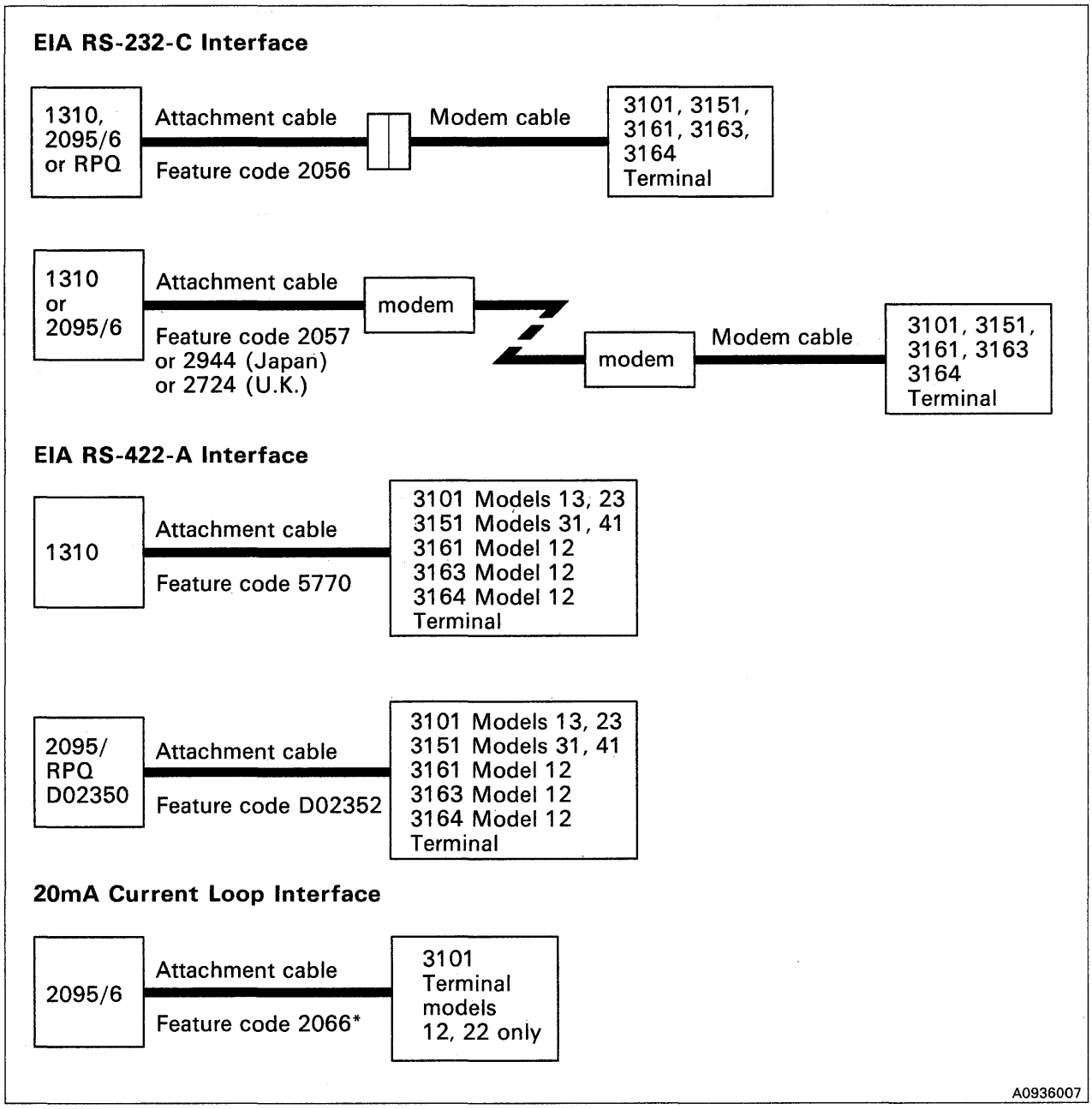


Figure C-15. 3101, 3151, 3161, 3163, and 3164 Terminal with #1310, #2095/#2096 and #2095/RPQ D02350

* This cable is modified according to the source of the transmit current. See Figure C-16 on page C-34 and Figure C-17 on page C-38.

Operator Input and Internal Code Considerations

For the 3101, 3151, 3161, 3163, or 3164 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 or F8 key.

When the 3101, 3151, 3161, 3163, or 3164 is connected through the #2095/#2096 attachment, pressing the PF8 key or F8 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, 3151, 3161, 3163, or 3164 is connected through the #1610 or #2091/#2092 attachment, pressing the PF8 key or F8 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).

Attaching Line-Sharing Devices

You can connect a Personal Computer with the Intelligent Workstation Programming RPQ (5799-TNG) and a 4201/4202 printer to the Series/1. Use attachment cable Feature Code #2056 (PN1632211) to connect the Series/1 #1310 or #2095/#2096 Feature Programmable Card to the asynchronous attachment card in the Personal Computer. Only port 0 of the #1310 Multifunction Attachment Feature has an RS-232-C interface. For remote attachment, connect the attachment card in the Series/1 to a modem using Cable Feature Code #2057.

Connect the 4201/4202 printer to the Personal Computer with a standard parallel-interface printer cable.

You can connect a 3151, 3161, 3163, or 3164 display terminal and a 4201/4202 printer to the Series/1. Connect the 3151, 3161, 3163, or 3164 display terminal to the Series/1 #1310 Multifunction Attachment Feature or #2095/#2096 Feature Programmable Card. See "Attaching 3101, 3151, 3161, 3163, and 3164 Display Terminals" on page C-23 for more information.

To attach the 4201/4202 printer to the 3151, 3161, 3163 or 3164 terminal, connect an ASCII terminal I/O cable (PN6343373) to the 3151, 3161, 3163 or 3164 terminal. Then connect a 4201/4202 attachment cable (PN8509386) to the ASCII terminal I/O cable and to the serial interface card (PN6493187) of the 4201/4202 printer.

As an alternative, you can connect the 3151, 3161, 3163 or 3164 terminal and the 4201/4202 serial interface card with a 25-pin male-to-male Electronic Industries Association (EIA) extender cable.

Configuring Your 3101, 3151, 3161, 3163, or 3164

To aid you in configuring your 3101, 3151, 3161, 3163, or 3164 display terminal, two figures are presented. Figure C-16 on page C-34 shows 22 configurations for the 3101 (in 3101 emulation only) in character mode; Figure C-17 on page C-38 shows 17 configurations for the 3101 in block mode.

The individual configurations are numbered sequentially starting with the first configuration in Figure C-16 and concluding with the configurations in Figure C-17. The configuration numbers appear in the left margin of the figure on the even-numbered pages and in the right margin on the odd-numbered pages of the charts. You can use these numbers when referring to a particular configuration.

Select the appropriate figure based upon the transmission mode (character or block) of your 3101, 3151, 3161, 3163, or 3164. 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:

- 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.

If you are using another terminal type 316x or 3151 terminal, 3101 terminal connected to a specific adapter type (1310, 7850, and so forth), match your setup menu switch settings to the comparable values shown for 3101 switch settings in relation to the given TERMINAL statements values.

Tell your IBM Customer Engineer which attachment options you require during Series/1 or attachment feature installation. He will install the appropriate options on the attachment feature card. You must specify the appropriate TERMINAL definition statement and set the 3101 switches, or the 3151, 3161, 3163, and 3164 setup switches according to the type of configuration.

While the configurations shown in Figure C-16 and Figure C-17 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.

	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-isolated (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 C-16 (Part 1 of 4). Configuration Matrix for 3101, in Character Mode

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=0A,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 C-16 (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	1310 MFA	58	–	–	–	–	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(18.1)		Full duplex	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	1310 MFA	58	–	–	–	–	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(22.1)		Full duplex	1310 MFA	58	–	–	–	–	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744

Figure C-16 (Part 3 of 4). Configuration Matrix for 3101, in Character Mode

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, BITRATE=1200,RANGE=HIGH	C	(15)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=60,MODE=3101C, BITRATE=1200,ADAPTER=FOUR, RANGE=HIGH	C C	(16)
Modem	5640736	10,12,13, 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=68,MODE=3101C CODTYPE=ASCII,BITRATE=1200,ATTN=1B68, LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, RANGE=HIGH	C C	(17)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58, ADAPTER=MFA,MODE=3101C	C	(18)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58, ADAPTER=MFA,MODE=3101C LMODE=SWPRTS	C C	(18.1)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=08,MODE=3101C, BITRATE=1200,LMODE=SWITCHED, RANGE=HIGH	C C	(19)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=60,MODE=3101C, BITRATE=1200,LMODE=SWITCHED, ADAPTER=FOUR,RANGE=HIGH	C C	(20)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=68,MODE=3101C, CODTYPE=ASCII,BITRATE=1200,ATTN=1B68, LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, RANGE=HIGH,LMODE=SWITCHED	C C C	(21)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58,MODE=3101C, LMODE=SWITCHED,ADAPTER=MFA	C	(22)
Modem	5640736	10,12,13 20,22,23	0111 0010	0000 0000	1011 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58,MODE=3101C, LMODE=SWPRTS,ADAPTER=MFA	C	(22.1)

Figure C-16 (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 C-17 (Part 1 of 4). Configuration Matrix for 3101, in Block Mode

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, BITRATE=9600,RANGE=HIGH	C (23)
Direct	5640736	20,22,23	1111 0010	0001 0000	1000 0000	1001 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58,MODE=3101B, BITRATE=9600,ADAPTER=MFA,LMODE=LOCAL	C (24)
Direct	5640736	20,22,23	1111 0010	0001 0000	1000 0000	1001 1001	TERMINAL	DEVICE=ACCA,ADDRESS=60,MODE=3101B, BITRATE=2400,ADAPTER=FOUR, RANGE=HIGH	C C (25)
Direct	5640736	20,22,23	1111 0010	0001 0000	1000 0000	1001 1001	TERMINAL	DEVICE=ACCA,ADDRESS=68,MODE=3101B, CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, RANGE=HIGH	C C C (26)
Direct	–	13,23	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58,MODE=3101B, BITRATE=9600,ADAPTER=MFA, LMODE=RS422	C C (27)
Direct	–	13,23	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL	DEVICE=ACCA,ADDRESS=68,MODE=3101B, BITRATE=9600,ADAPTER=EIGHT, LMODE=LOCAL,CODTYPE=ASCII	C C (28)
Direct	–	22	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL	DEVICE=ACCA,ADDRESS=68,MODE=3101B, CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, RANGE=HIGH	C C C (29)
Direct	–	22	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL	DEVICE=ACCA,ADDRESS=68,MODE=3101B, CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, RANGE=HIGH	C C C (30)
Direct	–	22	1101 0010	0001 0000	1000 0000	1001 1001	TERMINAL	DEVICE=ACCA,ADDRESS=68,MODE=3101B, CODTYPE=ASCII,BITRATE=9600,ATTN=1B68, LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, RANGE=HIGH	C C C (31)

Note: When coding the MODE= operand for the 3161, 3163, and 3164 in their own block mode, see "ACCA-Type Terminals" on page A-89 for the valid options.

Figure C-17 (Part 2 of 4). Configuration Matrix for 3101, in Block 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	
(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	1310 MFA	58	–	–	–	–	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(33.1)		Full duplex	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	1310 MFA	58	–	–	–	–	–	–	–	2057 or 2944 (Japan) 2724 (U.K.)	1632208 1632919 1727744
(37.1)		Full duplex	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 C-17 (Part 3 of 4). Configuration Matrix for 3101, in Block Mode

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, BITRATE=1200,RANGE=HIGH	C	(32)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58,MODE=3101B, ADAPTER=MFA	C	(33)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58,MODE=3101B, ADAPTER=MFA,LMODE=SWPRTS	C	(33.1)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=60,MODE=3101B, BITRATE=1200,ADAPTER=FOUR, RANGE=HIGH	C C	(34)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=68,MODE=3101B, CODTYPE=ASCII,BITRATE=1200,ATTN=1B68, LF=0A,CR=0D,PF1=1B61,ADAPTER=FOUR, RANGE=HIGH	C C C	(35)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=08,MODE=3101B, BITRATE=1200,LMODE=SWITCHED, RANGE=HIGH	C C	(36)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58,MODE=3101B, LMODE=SWITCHED,ADAPTER=MFA	C	(37)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=58,MODE=3101B, LMODE=SWPRTS,ADAPTER=MFA	C	(37.1)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=60,MODE=3101B, BITRATE=1200,LMODE=SWITCHED, ADAPTER=FOUR,RANGE=HIGH	C C	(38)
Modem	5640736	20,22,23	1111 0010	0001 0000	1000 0000	0101 1001	TERMINAL	DEVICE=ACCA,ADDRESS=68,MODE=3101B, CODTYPE=ASCII,BITRATE=1200, LMODE=SWITCHED,ATTN=1B68,LF=0A, CR=0D,PF1=1B61,ADAPTER=FOUR, RANGE=HIGH	C C C C	(39)

Figure C-17 (Part 4 of 4). Configuration Matrix for 3101, in Block Mode

Configuration Matrix Notes

1. The device addresses shown in Figure C-16 on page C-34 and Figure C-17 on page C-38 are typical. For additional information on device addresses, refer to the *IBM Series/1 Configurator*, GA34-0042.

   Note: The attachment feature device address jumpered on the attachment card must agree with the ADDRESS= operand of the TERMINAL definition statement specified during system generation. For a description of the TERMINAL definition statement, see Appendix A, "System Definition Statements."

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	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 C-18. Speed Range Jumper Options

Note: The bit rate of 19200 for the #2096 Feature-Programmable Communication Adapter feature is only supported for the 3151 and 316x terminals.

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= operand 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.

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 19200 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, refer to the *IBM Series/1 Customer Site Preparation Manual*, GA34-0050.

4. For additional information regarding adapter jumper requirements, refer to the *Machine Logic Diagrams* or the *IBM Series/1 Communication Features Theory Diagrams*, SY34-0059.

5. The cable part numbers referred to in Figure C-14 on page C-30 and Figure C-15 on page C-31 are subject to change. 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, refer to 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.

   Note: Set the 3151/ 3161/3163/3164 software switches to correspond to the 3101 using Figure C-4 on page C-12 and Figure C-6 on page C-15.

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 C-19. 3101 Supported Baud Rates and Group 4 Switch Settings

For additional information regarding the 3101 setup switches and interface support provided for the 3101, refer to the *IBM 3101 Display Terminal Description*, GA18-2033.

9. For a description of the TERMINAL definition statement, see the Appendix A, "System Definition Statements."

   a. For illustrative purposes, the RANGE= operand in the 3101 configuration matrices is coded as HIGH to emphasize that the RANGE= operand 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 system generation time by using the TIMERS operand 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 to) 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.

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.

The 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
  •
  •
  •
```

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, refer to the *IBM Series/1 Asynchronous Communications Feature Description*, GA34-0243.

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.

Entry Point	Object Module
$$SRTBL	EDXSYS
$$SSNA	EDXSYS
$A	RLOADER
$ACCARAM	$ACCRAM1
$ACCATRC	$ACCATRC
$ACTIVE	RLOADER
$ADPINIT	INITMODS
$ADPOPEN	INITMODS
$APPCCBS	EDXSYS
$APPCMAB	EDXSYS
$AUTOINT	INITMODS
$AUTOLOG	INITMODS
$AUTOSML	INITMODS
$BITS	EDXALU
$BLOCKCT	$EDXDEF
$BONOFF	EDXALU
$BSCADDR	EDXSYS
$BSCADS	XPSPART
$BSCAINT	INITMODS
$BSCAM	$BSCAM
$BSCARAM	$BSCARAM
$BSCFDDB	$EDXDEF (BSCLINE)
$BUFF	$EDXTIO
$CANCEL	RLOADER
$CFATTN	$EDXTIO
$CFEXTEN	INITMODS
$CFINIT1	INITMODS
$CFINIT2	INITMODS
$CFPARM	EDXSYS
$CMDINIT	EDXINIT1
$CN	RLOADER
$COADDR	EDXSYS
$COINIT	INITMODS
$COMBASE	$EDXDEF
$COMPADR	EDXALU
$COMPCNT	EDXALU
$COMPE	EDXALU
$COMPNE	EDXALU
$COMTABL	SWAITM
$COPAM	$COPAM
$CP	EDXSVCX
$CPBADDR	EDXSYS
$CPBINIT	INITMODS
$CSXINIT	INITMODS
$CVTADS	XPSPART
$DBUGNUC	$DBUGNUC
$DDBC	$EDXDEF
$DDBFIX	EDXINIT1
$DDBFIXR	EDXINIT1
$DECR1	SRCHK
$DECR2	SRCHK

Entry Point	Object Module
$DIDAXPS	$DIDAXPS
$DIDSKA	$DIDSKA
$DISKINT	INITMODS
$DISKIO$	$DISKIO
$DMDDB	EDXSYS
$DMVOL	EDXSYS
$DPEND	$EDXTIO
$DSKADS	XPSPART
$DSKINT1	INITMODS
$DSKINT2	INITMODS
$DSKXINT	INITMODS
$DSKXPS	$DSKXPS
$DYNEND	DYNEND
$DYNSTRT	DYNSTART
$DYNSTR1	DYNSTRT1
$D1024	$D1024
$D49624	$D49624
$D4963A	$D4963A
$D4966A	$D4966A
$D4969A	$D4969A
$D624XPS	$D624XPS
$D63AXPS	$D63AXPS
$D66AXPS	$D66AXPS
$D69AXPS	$D69AXPS
$EDXFLT	EDXFLT
$EDXINIT	EDXINIT1
$EDXTIO	$EDXTIO
$EDXTIO	NOTIO
$EXADDR	EDXSYS
$EDXPTCH	$EDXDEF
$EXEC	CMDPRT1
$EXEC	EDXALU
$EXIODDB	$EDXDEF (EXIO)
$EXIOINT	INITMODS
$FINDE	EDXALU
$FINDNE	EDXALU
$FLTADS	XPSPART
$FRSTACB	EDXSYS
$FRSTLCC	EDXSYS
$FSADDR	EDXSYS
$HOSTINT	INITMODS
$IAMQCB	EDXSYS
$INCR	SRCHK
$INITMOD	$EDXDEF
$INITPRT	$EDXDEF
$INITSHR	INITMODS
$INITWTM	INITMODS
$IOSACCA	$IOSACCA
$IOSS1S1	$IOSS1S1
$IOSTTY	$IOSTTY
$IOSVIRT	$IOSVIRT
$IOS3101	$IOS3101

Entry Point	Object Module
$IOS4013	$IOS4013
$IOS4974	$IOS4974
$IOS4979	$IOS4979
$IOUS	$EDXDEF
$IO3101B	$IOS3101
$IO4975A	$IO4975A
$IPLDEBG	INITMODS
$IPLEND	EDXSYS
$IPLHOOK	EDXSYS
$IPLVOL	EDXSYS
$LCCAM	$LCCAM
$LCCINT	INITMODS
$LCCPTC	INITMODS
$LCCXPS	$LCCXPS
$LOADBUF	LOADBUFR
$LOADINT	INITMODS
$LOADIN2	INITMODS
$LOGPARM	EDXSYS
$LOGPART	$EDXDEF
$LOGTSK	SYSLOG
$MAPAREA	$EDXDEF
$MEMSIZE	$EDXDEF
$MFAINIT	INITMODS
$MFAOPEN	INITMODS
$MPBDCB	MPBUFFER
$MPBLOCK	MPBUFFER
$MSGBUF	FULLMSG
$MSGBUF	MINMSG
$NUMPART	$EDXDEF
$OCTSTUB	$OVLMGR0
$OVLAREA	$OVLMGR0
$OVLCNT	$OVLMGR0
$OVLCT	$OVLMGR0
$OVLDSCB	$OVLMGR0
$OVLEDL	$OVLMGR0
$OVLERR	$OVLMGR0
$OVLMGRO	$OVLMGR0
$OVLPAD	$OVLMGR0
$OVLRTN1	$OVLMGR0
$OVLRTN2	$OVLMGR0
$OVLSIZE	$OVLMGR0
$OVLSTOP	$OVLMGR0
$PARTSZE	$EDXDEF
$PGMCINT	INITMODS
$PHYSTG	$EDXDEF
$PRINT2	$EDXTIO
$QIOADS	XPSPART
$SBAI	$SBAI
$SBCOM	$SBCOM
$SBDIDO	$SBDIDO
$SBIOADS	XPSPART
$SBIOINT	INITMODS
$SBPITAB	EDXSYS
$SDLC	EDXSYS
$SEGINIT	EDXINIT1
$SEGRET	EDXINIT1
SETRET	INITMODS
$SLOGIA	SYSLOGIA
$SLOGPIA	SYSLOGIA
$SFLIST	EDXSYS
$SLOGPRM	SYSLOG
$SLOGTSK	SYSLOG
$SNASTUB	#SNASTUB
$SRINIT1	INITMODS
$SRINIT2	INITMODS
$SRMADS	XPSPART
$SRMGR	$SRMGR
$SROPEN	INITMODS
$SRTBL	NOSRMGR
$SRTBL	$SRMGR
$START	EDXSYS
$STORAGE	$EDXDEF
$STORINT	INITMODS
$SUPVIO	SUPVIO
$SVCBCTL	EDXSYS
$SVCIBUF	$EDXDEF
$SVCLSB	EDXSVCX
$SVCSIA	EDXSVCX
$SVCSTOP	EDXSVCX
$SYSCOM	$EDXDEF
$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
$TIOXPS	$TIOXPS
$TPDDB	EDXSYS
$TPDDB1	TPCOM
$TPDVADR	$EDXDEF
$TPIA	TPCOM
$TRCLSB	EDXSTART
$TRCSIA	$DBUGNUC
$TRMINIT	INITMODS
$TRMOPEN	INITMODS
$USRATT1	$EDXTIO
$USRATT2	$EDXTIO
$USRATT3	$EDXTIO

Entry Point	Object Module
$USR1	ALUPRT1
$USR2	ALUPRT1
$VIRPART	EDXSYS
$XPSBCTL	EDXSYS
$XPSHEAD	EDXSYS
$XPSINIT	EDXINIT1
$XPSTABL	EDXSYS
$XTSK	X21CMD
$X21CVP	EDXSYS
$X21IA	$X21IA
$4013INT	INITMODS
$4978INT	INITMODS
$4980INT	INITMODS
$4980OPN	INITMODS
##ZERO##	EDXSYS
##ZERO##	XPSSYS
#ACLOSE	D4969A
#ALLSCB	NOSRMGR
#ALLSCB	SRMGR
#ATTNIDA	DIDSKA
#ATTNID1	DIDSKA
#ATTNID2	DIDSKA
#ATTN10	D49624
#ATTN10	D49624
#ATTN10A	D4963A
#ATTN35	D49624
#ATTN40	D49624
#ATTN50	D4966A
#ATTN51	D4966A
#ATTN55A	D4966A
#ATT1024	$D1024
#BMES	$SRMGR
#BSCENTR	BSCAM
#BSCIA	BSCAM
#BSCIORT	BSCAM
#BSCSTRT	BSCAM
#CMDSETU	EDXALU
#CMDSET1	CMDPRT1
#CMDSTUP	EDXALU
#CNTLEND	D4963A
#COIO	COPAM
#CSXRET	EDXINIT1
#DELRT	DISKIO
#DELSCB	NOSRMGR
#DELSCB	SRMGR
#DEQBSC	BSCAM
#DEQLBNK	NOLRDCHK
#DEQLBNK	LRDCHK
#DISKCHN	DISKIO
#DISKRW	DISKIO
#DISKTBL	DISKTBL
#DMGMT	DISKIO
#DMXTNT	DISKIO
#DMSTNT2	DISKIOX
#DQTERMS	EDXTERMQ
#DQTERMS	NOTIO
#DQTRMIN	EDXTERMQ
#DQTRMIN	NOTIO
#DSKCAC	D4963A
#DSKER13	DISKIO
#DSKIO00	DISKIO
#DSKIO99	$DISKIO
#DSKPOST	$DISKIO
#ENQLBNK	NOLRDCHK
#ENQLBNK	LRDCHK
#EDXFLOA	EDXFLOAT
#FLIHID	DIDSKA
#FLIHPST	$D4969A
#FLIH04	$DISKIO
#FLIH36A	D49624
#FLIH40	D4963A
#FLIH50	D49624
#FLIH54	D49624
#IAVIRT	IOSVIRT
#IDADCB	DIDSKA
#IDATTN	DIDSKA
#IDERP	DIDSKA
#IDINIT	DIDSKA
#IDSKCAC	DIDSKA
#IFB	EDXALU
#IFDW	EDXALU
#IFTEST	EDXALU
#IFTEST1	EDXALU
#IFW	EDXALU
#INITRET	EDXINIT1
#IOGPIB	IOSGPIB
#IOSGPIB	IOSGPIB
#IOSHARE	IOSHARE
#IOSPEND	IOSPOOL
#IOSPOOL	IOSPOOL
#IOSPWR	IOSPOOL
#IOSS1S1	IOSS1S1
#IOSTERM	IOSTERM
#IOSVIRT	IOSVIRT
#IOS2741	IOS2741
#IOS3101	IOS3101
#IOS316X	IOS316X
#IOVIRT	IOSVIRT
#IO4971	IOS4974
#IO4973	IOS4974
#IO4974	IOS4974
#IO4975	IOS4974
#IO4978	IOS4979
#IO4979	IOS4979

Entry Point	Object Module	Entry Point	Object Module
#IO5219	IOS4974	ACCATRC	ACCATRC
#IO5224	IOS4974	ACLOSE	D4969A
#IO5225	IOS4974	ACLOSE	$D69AXPS
#KBTASK	EDXTIO	ACLOSERT	RLOADER
#LCCDTCH	$LCCAM	AIIA	$SBAI
#LCCENTR	LCCAM	ALLSCB	NOSRCHK
#LCCSLIH	LCCAM	ALLSCB	SRCHK
#MESSAGE	FULLMSG	ALLSCB	SRCHK2
#MESSAGE	MINMSG	ALPOSTED	EDXSVCX
#MSGBUF	FULLMSG	ALUADDR	XPSTABLE
#NOP	EDXALU	ALUADS	XPSTABLE
#NXTCMDA	$EDXTIO	ALUBSC	ALUBSC
#PRTEXT	EDXTIO	ALUCF	ALUCF
#PWRAM80	PWRAM80	ALUDBUG	ALUDBUG
#QUEUEIO	QUEUEIO	ALUDSK	ALUDSK
#QUTRMIN	EDXTERMQ	ALUEXIO	ALUEXIO
#RDGPIB	IOSGPIB	ALUFLT	ALUFLT
#SBCOM	SBCOM	ALULCC	ALULCC
#SBCOM	CMDSIO	ALUPRT1	ALUPRT1
#SBPI	SBPI	ALUQIO	ALUQIO
#SLOGTSK	SYSLOG	ALURLDR	ALURLDR
#STATS78	IOS4979	ALUSIO	ALUSIO
#TAPEEND	$D4969A	ALUSTMGR	ALUSTMGR
#TERMCTL	NOTIO	ALUTIMER	ALUTIMER
#TERMCTL	IOS4224	ALUTIO	ALUTIO
#TERMCTL	NO4224	ALUTPCOM	ALUTPCOM
#TERMINT	EDXTIO	AND1	EDXALU
#TERMINT	NOTIO	AND1XX	EDXALU
#TERMOUT	EDXTIO	AND2	EDXALU
#TERMOUT	NOTIO	AND2XX	EDXALU
#TIOINST	CMDTIO	AND4	EDXALU
#TPEIO00	D4969A	AND4XX	EDXALU
#TPEIO99	$D4969A	ASCIITAB	TRASCII
#WHERES	RLOADER	ASMDEBUG	EDXINIT1
#WRGPIB	IOSGPIB	ATTACH	EDXSVCX
#2K	$SRMGR	ATTACHX	EDXSVCX
#4966B	D4966A	AUTOLOG	LOGLOAD
#624ATTN	D49624	AUTOSML	SMLLOAD
#624DCB	D49624	BDCDWORD	EDXTIO
#624ERP	D49624	BDCWORD	EDXTIO
#624INIT	D49624	BETWEEN	IOS4975A
#63ADCB	D4963A	BFRCK	EDXALU
#63ATTN	D4963A	BFRCK	CMDALU
#63ERP	D4963A	BHXWORD	EDXTIO
#63INIT	D4963A	BMETBL	$SRMGR
#66ADCB	D4966A	BME0	$SRMGR
#66AERP	D4966A	BME1	$SRMGR
#66ATTN	D4966A	BME2	$SRMGR
#69ATTN	D4969A	BME3	$SRMGR
#69ERP	D4969A	BME4	$SRMGR
#69ERP8	D4969A	BME5	$SRMGR
ACCACTL	IOSACCA	BME6	$SRMGR

Entry Point	Object Module
BME7	$SRMGR
BRACKN	$EDXTIO
BRANCH	EDXALU
BSCAM	$BSCAM
BSCCHK	BSCCHK
BSCENTRY	ALUBSC
BSCENTRY	BSCAM
BSCGRET	BSCAM
BSCGSCB	BSCCHK
BSCIA	BSCAM
BSCINIT	BSCINIT
BSCRETER	BSCAM
BSCX21	BSCX21
CCBFIX	CCBFIX
CCBFIXA	EDXINIT
CDIDQCB	IAMQCB
CDYSVC	CDYSVC
CDYSVCT	CDYSVC
CDRVTA	$EDXDEF (TERMINAL)
CDRVTB	$EDXDEF (TERMINAL)
CDXQAPND	CDXQAPND
CDXQPULL	CDXQPULL
CDXQPUSH	CDXQPUSH
CDXQWAIT	CDXQWAIT
CGOTO	EDXALU
CIRCBUFF	CIRCBUFF
CIRCNT	CIRCBUFF
CIREND	CIRCBUFF
CIRESIZ	CIRCBUFF
CIRESTR	CIRCBUFF
CIRIN	CIRCBUFF
CIRSTR	CIRCBUFF
CKEXIT	SBCOM
CKEXIT0	$SBCOM
CLOKINIT	CLOKINIT
CMDALU	CMDALU
CMDENTRY	CMDINIT
CMDINIT	CMDINIT
CMDPRT1	CMDPRT1
CMDRET	EDXINIT1
CMDSETUP	CMDALU
CMDSETUP	EDXALU
CMDSIO	CMDSIO
CMDTABLE	CMDTBLX
CMDTABLE	EDXSYS
CMDTADDR	EDXINIT1
CMDTBLX	CMDTBLX
CMDTIO	CMDTIO
CNTLBUSY	D4963A
CNTLEND	D4963A
CNTLEND	$D63AXPS
COCLEAN	COCLEAN

Entry Point	Object Module
COINIT	COINIT
COIO	COIO
CONINSIA	EDXSTART
COPAM	COPAM
COPCHK	COPCHK
COPGSCB	COPCHK
COPGSCB	NOCOPCHK
COPIAR	$COPAM
CRSPTAB	TRCRSP
CSXRET	EDXINIT1
CTLXFER	ALUTIO
CTLXFER	EDXTIO
CTLXFER	NOTIO
CURCTL	ALUTIO
CURCTL	EDXTIO
CURCTL	NOTIO
CURWSCB	$SRMGR
CURWSEG	$SRMGR
DATEFMT	$EDXDEF (TIMER)
DATTN1DA	$DIDAXPS
DATTN1DA	DIDSKA
DATTN1D1	$DIDAXPS
DATTN1D1	DIDSKA
DATTN1D2	$DIDAXPS
DATTN1D2	DIDSKA
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
DCB3DATA	TPCOM1
DCMCSET0	$DBUGNUC
DCMCSET1	$DBUGNUC
DCMCSET2	$DBUGNUC
DDBFIX	DDBFIX
DECSCAN	EDXTIO

Entry Point	Object Module
DELRT	$DSKXPS
DELRT	DISKIO
DELSCB	NOSRCHK
DELSCB	SRCHK
DELSCB	SRCHK2
DELSCBX	NOSRCHK
DELSCBX	SRCHK
DELSCBX	SRCHK2
DEQ	EDXSVCX
DEQBSC	$BSCAM
DEQHOST	TPCOM
DEQLBNK	LDRCHK
DEQLBNK	NOLDRCHK
DEQT	ALUTIO
DEQT	EDXTERMQ
DEQT	NOTIO
DETACH	EDXSVCX
DFLIHID	$DIDAXPS
DFLIHID	DIDSKA
DFLIH04	$DISKIO
DFLIH04	DSKXPS
DFLIH36A	$D624XPS
DFLIH36A	D49624
DFLIH40	D4963A
DFLIH40	$D63AXPS
DFLIH50	D49624
DFLIH50	$D624XPS
DFLIH54	D49624
DFLIH54	$D624XPS
DIDATTN	$DIDAXPS
DIDATTN	DIDSKA
DIDERP	$DIDAXPS
DIDERP	DIDSKA
DIDINIT	$DIDAXPS
DIDINIT	DIDSKA
DIDSKA	DIDSKA
DIDSKAT	$DIDSKA
DIDSKATN	DIDSKA
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

Entry Point	Object Module
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
DISKINIT	DISKINT1
DISKIO	$DSKXPS
DISKIO	DISKIO
DISKIOX	DISKIOX
DISKIO00	DISKIO
DISKIO00	$DSKXPS
DISKIO32	D4966A
DISKIO40	D4966A
DISKPOST	$DISKIO
DISKPOST	DSKXPS
DISKPREP	DISKINIT
DISKRET	DISKIO
DISKRET2	DISKIO
DISKRW	ALUDSK
DISKRW	DISKIO
DISKRW	$DSKXPS
DISKRW05	DISKIO
DISKRW09	DISKIO
DISKTBL	DISKTBL
DISKXCOM	DISKIOX
DISKXDEQ	DISKIOX
DISKXINT	DSKXINIT
DISP	$SRMGR
DLCROUTE	DLCROUTE
DMDDB	$EDXDEF (DISK)
DMEXBUFR	DMEXBUFR
DMEXTBL	DMEXTBL
DMIPL	$EDXDEF (DISK)
DMVOL	$EDXDEF (DISK)
DMXBUFFR	DMEXBUFR
DMXTNT	ALUDSK
DMXTNT	DISKIO
DMXTNT2	DISKIOX
DMXTNT3	DISKIO
DMXTNT3	DISKIOX
DOIA	$SBDIDO
DQTERM	EDXTERMQ

Entry Point	Object Module
DQTERMB	EDXTERMQ
DQTERMS	EDXTERMQ
DQTRMIN	EDXTERMQ
DSKCAC	D4963A
DSKCAC	$D63AXPS
DSKCHKA	DSKCHK
DSKCHKA	NODSKCHK
DSKCHKAR	DISKIO
DSKCHKD	DSKCHK
DSKCHKD	NODSKCHK
DSKCHKDR	DISKIO
DSKCHKER	DISKIO
DSKCHKX	DSKCHK
DSKINITX	DSKINIT2
DSKINIT1	DSKINIT1
DSKINIT2	DSKINIT2
DSKPREP2	DISKINIT
DSKXINIT	DSKXINIT
DSKXPS	DSKXPS
DSKXRET1	DISKIO
DSKXRET2	DISKIO
DSKXRET3	DISKIO
DYNEND	DYNEND
DYNSTART	DYNSTART
DYNSTRT1	DYNSTRT1
D1024	D1024
D1024ATN	D1024
D1024FLG	$D4966A
D1024R	D1024
D1024RET	D1024
D1024V1	$D1024
D4962IH1	D49624
D49624	D49624
D49624AT	$D49624
D4963A	D4963A
D4963AT	$D4963A
D4963ATN	D4963A
D4963AT1	D4963A
D4963IH1	D4963A
D4966A	D4966A
D4966AT	$D4966A
D4966ATN	D4966A
D4966B	D4966A
D4966B	$D66AXPS
D4969A	D4969A
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
EBASCII	TREBASC
EBBICVT	ALUTIO
EBBICVT	EDXTIO
EBBICVT	NOTIO
EBBIE	EDXTIO
EBCDTAB	TREBCD
EBFLCVT	EBFLCVT
EBFLDBL	EBFLCVT
EBFLPTCH	EBFLCVT
EBFLSTD	EBFLCVT
EDXALU	EDXALU
EDXDEBUG	EDXINIT1
EDXFLAGS	EDXSYS
EDXFLEND	ALUFLT
EDXFLEND	EDXFLOAT
EDXFLOAT	EDXFLOAT
EDXINIT	EDXINIT
EDXINIT1	EDXINIT1
EDXLOOP	EDXALU
EDXSTART	EDXSTART
EDXSVCX	EDXSVCX
EDXSYS	EDXSYS
EDXTERMQ	EDXTERMQ
EDXTIMER	EDXTIMER
EDXTIMR2	EDXTIMR2
EDXTIO	EDXTIO
ENABLED	EDXINIT1
ENDATTN	EDXTIO
ENDATTN	ALUTIO
ENDCODE	RLOADER
ENDINIT	EDXSTART
ENQ	EDXSVCX
ENQLBNK	LDRCHK
ENQLBNK	NOLDRCHK

Entry Point	Object Module
ENQT	ALUTIO
ENQT	EDXTERMQ
ENQT	NOTIO
EOR1	EDXALU
EOR1XX	EDXALU
EOR2	EDXALU
EOR2XX	EDXALU
EOR4	EDXALU
EOR4XX	EDXALU
ERAPCMD	SYSLOG
ERREXIT3	RLOADER
ERRORLOG	DISKIO
EXCLOSE	IOSEXIO
EXFLIH	IOSEXIO
EXIO	IOSEXIO
EXIO	ALUEXIO
EXIOCLEN	IOSEXIO
EXIOINIT	EXIOINIT
EXIOTRC	EXIOTRC
EXOPEN	IOSEXIO
EXOPEN	ALUEXIO
EXTSTRT	DMEXTBL
FADD000	ALUFLT
FADD000	EDXFLOAT
FADD000	NOFLOAT
FADD001	ALUFLT
FADD001	EDXFLOAT
FADD001	NOFLOAT
FADD010	ALUFLT
FADD010	EDXFLOAT
FADD010	NOFLOAT
FADD011	ALUFLT
FADD011	EDXFLOAT
FADD011	NOFLOAT
FADD100	ALUFLT
FADD100	EDXFLOAT
FADD100	NOFLOAT
FADD101	ALUFLT
FADD101	EDXFLOAT
FADD101	NOFLOAT
FADD110	ALUFLT
FADD110	EDXFLOAT
FADD110	NOFLOAT
FADD111	ALUFLT
FADD111	EDXFLOAT
FADD111	NOFLOAT
FDIV000	ALUFLT
FDIV000	EDXFLOAT
FDIV000	NOFLOAT
FDIV001	ALUFLT
FDIV001	EDXFLOAT
FDIV001	NOFLOAT
FDIV010	ALUFLT
FDIV010	EDXFLOAT
FDIV010	NOFLOAT
FDIV011	ALUFLT
FDIV011	EDXFLOAT
FDIV011	NOFLOAT
FDIV100	ALUFLT
FDIV100	EDXFLOAT
FDIV100	NOFLOAT
FDIV101	ALUFLT
FDIV101	EDXFLOAT
FDIV101	NOFLOAT
FDIV110	ALUFLT
FDIV110	EDXFLOAT
FDIV110	NOFLOAT
FDIV111	ALUFLT
FDIV111	EDXFLOAT
FDIV111	NOFLOAT
FH4963CT	RW4963ID
FIRSTACB	$EDXDEF (ADAPTER)
FIRSTCCB	$EDXDEF (TERMINAL)
FIRSTDYN	RLOADER
FLDCLEAR	EDXTIO
FLEBDBL	EBFLCVT
FLEBSTD	EBFLCVT
FLIHPOST	$D4969A
FLIHPOST	DSKXPS
FLOATERR	$EDXFLT
FLOATERR	ALUFLT
FLOATERR	EDXFLOAT
FLOATERR	NOFLOAT
FLRND	EBFLCVT
FLTCONV	ALUFLT
FLTCONV	EDXFLOAT
FLTCONV	NOFLOAT
FMPY000	ALUFLT
FMPY000	EDXFLOAT
FMPY000	NOFLOAT
FMPY001	ALUFLT
FMPY001	EDXFLOAT
FMPY001	NOFLOAT
FMPY010	ALUFLT
FMPY010	EDXFLOAT
FMPY010	NOFLOAT
FMPY011	ALUFLT
FMPY011	EDXFLOAT
FMPY011	NOFLOAT
FMPY100	ALUFLT
FMPY100	EDXFLOAT
FMPY100	NOFLOAT
FMPY101	ALUFLT
FMPY101	EDXFLOAT

Entry Point	Object Module
FMPY101	NOFLOAT
FMPY110	ALUFLT
FMPY110	EDXFLOAT
FMPY110	NOFLOAT
FMPY111	ALUFLT
FMPY111	EDXFLOAT
FMPY111	NOFLOAT
FREEMAIN	RLOADER
FREEMAIN	ALURLDR
FREERETN	STORMGR
FREESTGC	STORMGR
FRSTSCB	$SRMGR
FSUB000	ALUFLT
FSUB000	EDXFLOAT
FSUB000	NOFLOAT
FSUB001	ALUFLT
FSUB001	EDXFLOAT
FSUB001	NOFLOAT
FSUB010	ALUFLT
FSUB010	EDXFLOAT
FSUB010	NOFLOAT
FSUB011	ALUFLT
FSUB011	EDXFLOAT
FSUB011	NOFLOAT
FSUB100	ALUFLT
FSUB100	EDXFLOAT
FSUB100	NOFLOAT
FSUB101	ALUFLT
FSUB101	EDXFLOAT
FSUB101	NOFLOAT
FSUB110	ALUFLT
FSUB110	EDXFLOAT
FSUB110	NOFLOAT
FSUB111	ALUFLT
FSUB111	EDXFLOAT
FSUB111	NOFLOAT
FULLMSG	FULLMSG
GETCNT	EDXALU
GETDDB	SBCOM
GETMAIN	ALURLDR
GETMAIN	RLOADER
GETPAR3	CMDALU
GETPAR3	EDXALU
GETRETF	STORMGR
GETRETN	STORMGR
GSTVAL2	ALUTIO
GETVAL2	EDXTIO
GSTVAL2	NOTIO
GSTVAL4	ALUTIO
GETVAL4	EDXTIO
GSTVAL4	NOTIO
GFQCB	RLOADER
GPIB	$EDXDEF
GTIMDATE	ALUTIMER
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
HCRESP0	TPCOM1
HCRESP1	TPCOM1
HCRETCD	TPCOM1
HCSMID	TPCOM1
HCSTBYTE	TPCOM1
HCTDBUFF	TPCOM1
HCWACK	TPCOM1
IAACCA	$IOSACCA
IAACCATR	$ACCATRC
IAEXIOTR	EXIOTRC
IAGPIB	$IOSGPIB
IAMQCB	CDIDQCB
IAMQCB	IAMQCB
IAPACING	$IO4975A
IAPROC	$1OS2741
IAS1S1	$IOSS1S1
IATTY	$IOSTTY
IA2741	$IOS2741
IA4013	$IOS4013
IA4971	$IOS4974
IA4973	$IOS4974
IA4974	$IOS4974
IA4975	$IOS4974
IA4978	$IOS4979
IA4979	$IOS4979
IA4980	$IOS4979
IA5219	$IOS4974
IA5224	$IOS4974
IA5225	$IOS4974
IDCBRES	TPCOM1
IDCBSCSS	TPCOM1
IDCBSIO	TPCOM1
IDCBSIOD	TPCOM1
IDCBTOD	TPCOM1
IDSKCAC	DIDSKA

Entry Point	Object Module	Entry Point	Object Module
IDSKCAC	$DIDAXPS	IOSPCMD	$IOSPOOL
IDSKIO99	$DISKIO	IOSPDQT	IOSPOOL
IDSKIO99	DSKXPS	IOSPEND	$IOSPOOL
IDSKPOST	$DISKIO	IOSPNQT	IOSPOOL
IDSKPOST	DSKXPS	IOSPOOL	$IOSPOOL
ID4963IH	RW4963ID	IOSPTBL	$IOSPOOL
IFFLOAT	ALUFLT	IOSPWR	IOSPOOL
IFFLOAT	NOFLOAT	IOSRETN	IOSACCA
IFFLOATL	ALUFLT	IOSTABLE	IOSTABLE
IFFLOATL	NOFLOAT	IOSTBLE	IOSTABLE
INITADAP	INITADAP	IOSTERM	IOSTERM
INITEXIT	EDXINIT1	IOSTRT	IOSACCA
INITFEAT	EDXINIT1	IOSTTY	IOSTTY
INITMFA	INITMFA	IOS1S1	IOSS1S1
INITMODE	INITMODS	IOS2741	$IOS2741
INITMODS	INITMODS	IOS3101D	IOS3101
INITSEGS	INITMODS	IOS4013	IOS4013
INITSHAR	INITSHAR	IOS4224	IOS4224
INITTASK	EDXSTART	IOS4974	IOS4974
INITWTM	INITWTM	IOS4975A	IOS4975A
INIT4013	INIT4013	IOS4979	IOS4979
INIT4978	INIT4978	IOS4979M	IOS4979
INIT4980	INIT4980	IOUNLOAD	IOLOADER
INTIME	ALUTIMER	IOVIRT	$IOSVIRT
INTIME	EDXTIMER	IO1024	IO1024
INTIME	EDXTIMR2	IO1024I	IO1024
INTIMEX	ALUTIMER	IO1024RT	DISKINIT
INTIMEX	EDXTIMER	IO3101	IOS3101
INTIMEX	EDXTIMR2	IO316X	IOS316X
IOACPOST	ACCATRC	IO4974	$TIOXPS
IOACPOST	NOACCATR	IO4974	IOS4974
IOACPRE	ACCATRC	IO4975A	IOS4975A
IOACPRE	NOACCATR	IO4979	$TIOXPS
IOERROR	IOSACCA	IO4979	IOS4979
IOEXPOST	EXIOTRC	IPLENDED	EDXSTART
IOEXPOST	NOEXIOTR	IPLMSG	EDXINIT1
IOEXPRE	EXIOTRC	KBTASK	$TIOXPS
IOEXPRE	NOEXIOTR	KBTASK	EDXTIO
IOGPIB	$IOSGPIB	LASTACC	$SRMGR
IOLOAD	IOLOADER	LASTDYN	RLOADER
IOLOADER	IOLOADER	LASTSTAT	RLOADER
IOR1	EDXALU	LCB	SYSLOG
IOR1XX	EDXALU	LCBA	EDXSYS
IOR2	EDXALU	LCCAM	LCCAM
IOR2XX	EDXALU	LCCDTCH	$LCCAM
IOR4	EDXALU	LCCDTCH	LCCXPS
IOR4XX	EDXALU	LCCENTRY	ALULCC
IOSACCA	IOSACCA	LCCENTRY	LCCAM
IOSEXIO	IOSEXIO	LCCFLIH	$LCCAM
IOSGPIB	$IOSGPIB	LCCINT	LCCINT
IOSHARE	IOSHARE	LCCNEXT	LCCPTC
IOSPCLOS	IOSPOOL	LCCPTC	LCCPTC

Entry Point	Object Module
LCCSLIH	$LCCXPS
LCCSLIH	LCCAM
LCCXPS	LCCXPS
LCMDKEY	RLOADER
LCMDTGT	RLOADER
LCPOV1	LCPOV1
LCPOV2	LCPOV2
LDRCHK	LDRCHK
LEX	EDXSVCX
LOADBANK	$SRMGR
LOADBUFR	LOADBUFR
LOADEXIT	RLOADER
LOADFHFL	RLOADER
LOADFLAG	$SRMGR
LOADFLAG	NOSRMGR
LOADINIT	LOADINIT
LOADINT2	LOADINT2
LOADORG	RLOADER
LOADPGM	RLOADER
LOADPGM	ALURLDR
LOADPGM0	RLOADER
LOADQCB	RLOADER
LOADQCB2	$SRMGR
LOADTASK	RLOADER
LOADTERM	PWRAM80
LOGLOAD	LOGLOAD
LOOPCNT	$SRMGR
LPGMXPA	RLOADER
LPGMXPB	RLOADER
LPRINTER	$EDXDEF (TERMINAL)
LP0	RLOADER
MAPEND	$EDXDEF
MAXWSCB	$SRMGR
MAXWSEG	$SRMGR
MAX2K	$SRMGR
MECBLST	$EDXDEF
MESSAGE	ALUTIO
MESSAGE	FULLMSG
MESSAGE	MINMSG
MESSAGE	NOTIO
MFARMU	$EDXDEF (ADAPTER)
MGRADS	$SRMGR
MINMSG	MINMSG
MOVEXP	EDXALU
MOVFP4	ALUFLT
MOVFP4	EDXFLOAT
MOVFP4	NOFLOAT
MOVFP8	ALUFLT
MOVFP8	EDXFLOAT
MOVFP8	NOFLOAT
MOV1	EDXALU
MOV1C	EDXALU
MOV2	EDXALU
MOV2C	EDXALU
MOV4	EDXALU
MOV4C	EDXALU
MPBUFFER	MPBUFFER
MPRTR1	$EDXDEF (TERMINAL)
MPRTR2	$EDXDEF (TERMINAL)
MPRTR3	$EDXDEF (TERMINAL)
MSGBUFR	EDXSTART
MSGMOD	FULLMSG
MSGMOD	MINMSG
NAKMSG	$EDXTIO
NEXTERM	TERMINIT
NOACCATR	NOACCATR
NOBSCCHK	NOBSCCHK
NOCOPCHK	NOCOPCHK
NODSKCHK	NODSKCHK
NOEXIOTR	NOEXIOTR
NOFLOAT	NOFLOAT
NOLDRCHK	NOLDRCHK
NOSRCHK	NOSRCHK
NOSRMGR	NOSRMGR
NOSTRMGR	NOSTRMGR
NOTDEF	$EDXTIO
NOTIO	NOTIO
NOTPWRON	$IOS4979
NO4224	NO4224
NSRMGR	NOSRMGR
NUMPART	EDXSYS
NXTCOMD	EDXTIO
OPENADAP	OPENADAP
OPENMFA	OPENMFA
OPEN4980	OPEN4980
OPNCLS	D4969A
OVLEND	DISKINIT
OVLSTART	SEGINIT
PACK	RSLCMPR
PARTFLAG	$SRMGR
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

Entry Point	Object Module
PRIMSTRT	DMEXTBL
PRINTIME	ALUTIMER
PRINTIME	EDXTIMER
PRINTIME	EDXTIMR2
PRSKSP	ALUTIO
PRSKSP	EDXTIO
PRSKSP	NOTIO
PRTEXT	ALUTIO
PRTEXT	EDXTIO
PRTEXT	NOTIO
PRTNUM2	ALUTIO
PRTNUM2	EDXTIO
PRTNUM2	NOTIO
PRTNUM2S	EDXTIO
PRTNUM4	ALUTIO
PRTNUM4	EDXTIO
PRTNUM4	NOTIO
PRTNUM4S	EDXTIO
PSTUSER	DISKIO
PTCHTBL1	SRCHK
PTCHTBL1	SRCHK2
PTCHTBL2	SRCHK2
PTCHTBL2	SRINIT2
PWRAM80	PWRAM80
PWRM80	PWRAM80
PWRTEST	PWRAM80
QIO	ALUQIO
QIO	QUEUEIO
QUESTION	ALUTIO
QUESTION	EDXTIO
QUESTION	NOTIO
QUEUEIO	QUEUEIO
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	$IOSGPIB
RDS1S1	IOSS1S1
RDTEXT	ALUTIO
RDTEXT	EDXTIO
RDTEXT	NOTIO
RDTEXTL	ALUTIO
RDTEXTL	EDXTIO
RDTEXTL	NOTIO
RDTTY	IOSTTY
RD2741	IOS2741
RD4013	IOS4013
RECKSUM	RSLCMPR
RETURN	EDXSYS
RLENGTH	TPCOM1
RLOADER	RLOADER
RW4963ID	RW4963ID
S$CMDTB	ALUCF
S$EXEC	ALUCF
S$LCCADR	LCCPTC
S$LCCECB	LCCPTC
S$LCDDB	LCCPTC
S$PTCDSN	LCCPTC
S$PTCHDS	LCCPTC
SAI	ALUSIO
SAI	SBAI
SAIA	ALUSIO
SAIA	SBAI
SAIS	ALUSIO
SAIS	SBAI
SAIX	ALUSIO
SAIX	SBAI
SAO	ALUSIO
SAO	SBAO
SAOA	ALUSIO
SAOA	SBAO
SAOX	ALUSIO
SAOX	SBAO
SATTACH	ALUPRT1
SATTACH	EDXSVCX
SAVERET	EDXSVCX
SAVEXIT	EDXSVCX
SAV222CR	EDXALU
SAV224CR	EDXALU
SAV424CR	EDXALU
SAV444CR	EDXALU
SAX222	EDXALU
SA222	EDXALU
SA222C	EDXALU
SA424	EDXALU
SA424C	EDXALU
SBAI	SBAI
SBAO	SBAO
SBCOM	SBCOM
SBDIDO	SBDIDO
SBERR	SBCOM
SBIODDB	$EDXDEF (SENSORIO)
SBIOINIT	SBIOINIT
SBPI	$SBPI

Entry Point	Object Module
SCALL	EDXALU
SCBTBL	SCBTBL
SCB0	SCBTBL
SCONTINU	EDXALU
SCWAIT	ALUPRT1
SCWAIT	EDXSVCX
SDEQ	ALUPRT1
SDEQ	EDXSVCX
SDETACH	ALUPRT1
SDETACH	EDXSVCX
SDI	ALUSIO
SDI	SBDIDO
SDIA	ALUSIO
SDIA	SBDIDO
SDIS	ALUSIO
SDIS	SBDIDO
SDIX	ALUSIO
SDIX	SBDIDO
SDO	ALUSIO
SDO	SBDIDO
SDOA	ALUSIO
SDOA	SBDIDO
SDOLOOP	EDXALU
SDOP	ALUSIO
SDOP	SBDIDO
SDOS	ALUSIO
SDOS	SBDIDO
SDOX	ALUSIO
SDOX	SBDIDO
SD222	EDXALU
SD222C	EDXALU
SD422CR	EDXALU
SD422R	EDXALU
SD424	EDXALU
SD424C	EDXALU
SEGCNT	$SRMGR
SEGCNTX	$SRMGR
SEGINIT	SEGINIT
SEGMAX	$SRMGR
SEGREG	$SRMGR
SEGREGE	ALUSTMGR
SEGREGE	NOSTRMGR
SEGREGE	STORMGR
SELB01	EDXSVCX
SENQ	ALUPRT1
SENQ	EDXSVCX
SETBUSY	EDXSVCX
SETBUSY	XPSSYS
SETCLOCK	EDXTIMER
SETCLOCK	EDXTIMR2
SETIMER	ALUTIMER
SETIMER	EDXTIMER
SETIMER	EDXTIMR2
SETREADY	EDXSVCX
SFLIST	$EDXTIO
SFTKSIA	EDXSTART
SHIFTCNT	$SRMGR
SHL1	EDXALU
SHL1XX	EDXALU
SHL2	EDXALU
SHL2XX	EDXALU
SHL4	EDXALU
SHL4XX	EDXALU
SHR1	EDXALU
SHR1XX	EDXALU
SHR2	EDXALU
SHR2XX	EDXALU
SHR4	EDXALU
SHR4XX	EDXALU
SMLLOAD	SMLLOAD
SM222	EDXALU
SM222C	EDXALU
SM424	EDXALU
SM424C	EDXALU
SPOST	ALUPRT1
SPOST	EDXSVCX
SRCHK	SRCHK
SRCHK2	SRCHK2
SRESETEV	ALUPRT1
SRESETEV	EDXSVCX
SRETURN	EDXALU
SRINITX	SRINIT1
SRINITX2	SRINIT2
SRINIT1	SRINIT1
SRINIT2	SRINIT2
SRMGR	SRMGR
SRMGRFLG	$SRMGR
SROPEN	SROPEN
SROPENX	SROPEN
SSX222	EDXALU
SS222	EDXALU
SS222C	EDXALU
SS424	EDXALU
SS424C	EDXALU
START	EDXINIT
STARTFLG	EDXINIT
STARTPGM	EDXSTART
START1	EDXINIT1
START2K	$SRMGR
STATUS	IOSACCA
STESTIN	ALUDBUG
STESTIN	$DBUGNUC
STESTIN1	$DBUGNUC
STESTOUT	ALUDBUG

Entry Point	Object Module	Entry Point	Object Module
STESTOUT	$DBUGNUC	SUP$$RET	SUP$$RET
STKINIT	SEGINIT	SUP$$RET	NOS$$RET
STMTBLE	STORMGR	SUPEXIT	EDXSVCX
STOP	ALURLDR	SUPEXIT	XPSSYS
STOP	RLOADER	SUPEXTRL	EDXSVCX
STOPBA	SRINIT1	SUPLEX	EDXSVCX
STOPBB	SRINIT2	SUPLVLX0	EDXSVCX
STOPBC	SRINIT1	SUPTBL	EDXSVCX
STOPBE	SRINIT1	SUPVIO	SUPVIO
STOPBF	SRINIT1	SVC	EDXSVCX
STOPC0	SRINIT1	SVC	XPSSYS
STOPC1	SRINIT1	SVCA	EDXSVCX
STOPC2	SRINIT1	SVCAKR	EDXSYS
STOPC3	SRINIT1	SVCBF	$EDXDEF
STOPC4	TIMRINIT	SVCBFEND	$EDXDEF
STOPC5	BSCINIT	SVCBFIN	EDXSYS
STOPDA	INIT4980	SVCBFOUT	EDXSYS
STOPDB	INITMFA	SVCBOTH	EDXSVCX
STOPDC	INITMFA	SVCFLAGS	EDXSYS
STOPDD	INITMFA	SVCFLGS2	EDXSVCX
STOPDE	INITMFA	SVCHNDLR	EDXSVCX
STOPDF	INITMFA	SVCI	EDXSVCX
STOPD3	$OVLMGRO	SVCIAKR	EDXSYS
STOPEB	INIT4980	SVCIAR	EDXSYS
STOPEC	INITADAP	SVCIBFOF	EDXSVCX
STOPED	INITADAP	SVCIIAR	EDXSYS
STOPEE	INITADAP	SVCILSB	EDXSYS
STOPEF	INITADAP	SVCILSR	EDXSYS
STOPFA	SEGINIT	SVCIR0	EDXSYS
STOPFB	EDXSVCX	SVCIR1	EDXSYS
STOPFC	EDXSVCX	SVCIR2	EDXSYS
STOPFD	EDXSVCX	SVCIR3	EDXSYS
STOPFE	EDXSVCX	SVCIR4	EDXSYS
STOPFF	EDXALU	SVCIR5	EDXSYS
STOPF0	SEGINIT	SVCIR6	EDXSYS
STOPF1	XPSINIT	SVCIR7	EDXSYS
STOPF2	XPSINIT	SVCLSB	EDXSYS
STOPF3	XPSINIT	SVCLSR	EDXSYS
STOPF4	XPSINIT	SVCLT	EDXSYS
STOPF5	XPSINIT	SVCL1	EDXSYS
STOPF6	XPSINIT	SVCL2	EDXSYS
STOPF7	XPSINIT	SVCL3	EDXSYS
STOPF8	XPSINIT	SVCPARMS	EDXSYS
STOPF9	XPSINIT	SVCRTRN	EDXSYS
STOPTASK	RLOADER	SVCR0	EDXSYS
STOREMAP	$EDXDEF	SVCR1	EDXSYS
STORINIT	STORINIT	SVCR2	EDXSYS
STORMGR	STORMGR	SVCR3	EDXSYS
STP	ALUTPCOM	SVCR4	EDXSYS
STP	TPCOM	SVCR5	EDXSYS
STPTASK1	EDXSVCX	SVCR6	EDXSYS
STPTASK2	EDXSVCX	SVCR7	EDXSYS

Entry Point	Object Module
SVCTRC	EDXSVCX
SVC0	EDXSVCX
SVC1	EDXSVCX
SVC2	EDXSVCX
SWAIT	ALUPRT1
SWAIT	EDXSVCX
SWAITM	SWAITM
SX224CR	EDXALU
SX224R	EDXALU
SX444	EDXALU
SX444C	EDXALU
SYSCOM	EDXSYS
SYSLOGIA	SYSLOGIA
SYSLOG	SYSLOG
S1S1	$EDXDEF (TERMINAL)
S1S1DIAG	S1S1INIT
S1S1INIT	S1S1INIT
TAPEBUFR	TAPEBUFR
TAPEEND	$D4969A
TAPEEND	DSKXPS
TAPEINIT	TAPEINIT
TAPEIO	D4969A
TAPEIO00	D4969A
TAPEIO00	$D69AXPS
TAPEIO99	$D4969A
TAPEIO99	DSKXPS
TAPE060	DISKIO
TASKWD	EDXSTART
TBLCNTL	DMEXTBL
TBPARTL	$EDXDEF
TBUFFRA	TAPEBUFR
TERMADDR	XPSTABLE
TERMCTRL	ALUTIO
TERMCTRL	IOS4224
TERMCTRL	NOTIO
TERMCTRL	NO4224
TERMDEFS	$EDXDEF (TERMINAL)
TERMERRX	INIT4980
TERMERRX	TERMINIT
TERMINID	TERMINIT
TERMINIT	TERMINIT
TERMINT	EDXTIO
TERMOPEN	TERMOPEN
TERMOUT	EDXTIO
TIMERDDB	$EDXDEF (TIMER)
TIMERDDB	$EDXTIMR2
TIMER0	$EDXDEF (TIMER)
TIMER0	EDXTIMR2
TIMER0IA	EDXTIMER
TIMER0IA	EDXTIMR2
TIMER1	$EDXDEF (TIMER)
TIMER1IA	EDXTIMER

Entry Point	Object Module
TIMRINIT	TIMRINIT
TIMRINIT	CLOKINIT
TIMRLSB	EDXTIMR2
TLENGTH	TPCOM1
TLSB	EDXSVCX
TPCOM	TPCOM
TPCOM1	TPCOM1
TPIABUSY	$D4969A
TPIABUSY	DSKXPS
TPINIT	TPINIT
TPSTATS	TPCOM
TRACEXON	$IO4975A
TRASCII	TRASCII
TRCRSP	TRCRSP
TREBASC	TREBASC
TREBCD	TREBCD
TSOPEN	D4969A
UNCHAIN	EDXSVCX
UNCHAIN	XPSSYS
UNCHAKR1	EDXSYS
UNCHSAV6	EDXSYS
USER	EDXALU
USERPOST	DISKIO
VARYDSCB	$DISKIO
VARYEXIT	$DISKIO
VARYOFF	$DISKIO
VARYON	$DISKIO
VARYQCB	$DISKIO
VARYWORD	$DISKIO
VCTBL	IOSTABLE
VRYBUFF	TAPEBUFR
VRYIDSK	$DIDSKA
VRY4966	$D4966A
VRY4969	$D4969A
WAIT	EDXSVCX
WAITECB	$SRMGR
WAITIMER	ALUTIMER
WAITIMER	EDXTIMER
WAITIMER	EDXTIMR2
WAITM	SWAITM
WAITMR	SWAITM
WDCB1	TPCOM1
WDCB2	TPCOM1
WDCB3	TPCOM1
WDCB3CHN	TPCOM1
WDCB4	TPCOM1
WDCB4CHN	TPCOM1
WDCB5	TPCOM1
WHATIME	EDXTIMER
WHATIME	EDXTIMR2
WHATMSG	$EDXTIO
WHERES	RLOADER

Entry Point	Object Module
WHERES	ALURLDR
WORKAREA	WORKAREA
WORKSCB	SRMGR
WRACCA	IOSACCA
WRGPIB	$IOSGPIB
WRKSCBA	$SRMGR
WRS1S1	IOSS1S1
WRTTY	IOSTTY
WR2741	IOS2741
WR4013	IOS4013
WR4975A	IOS4975A
WTMERCD	EDXSVCX
XPSBAL1	EDXSVCX
XPSBAL6	EDXSVCX
XPSBAL7	EDXSVCX
XPSBR	EDXSVCX
XPSDEBUG	EDXSVCX
XPSHEADR	SEGINIT
XPSINIT	XPSINIT
XPSINITX	XPSINIT
XPSRET2	EDXSVCX
XPSRET3	EDXSVCX
XPSSTK	$EDXDEF
XPSSTKE	$EDXDEF
XPSSTK	$EDXDEF
XPSSTKOF	EDXSVCX
XPSSTKE	$EDXDEF
XPSSVCI	EDXSVCX
XPSSVCI	XPSSYS
XPSSVCIX	EDXSVCX
XPSSYS	XPSSYS
XPSTABLE	XPSTABLE
XPSTABLX	XPSTABLE
XPSTBL	XPSTABLE
XPSTSIZE	XPSTABLE
X21CLR	BSCX21
X21CMD	X21CMD
X21CON	BSCX21

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 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 4-1. 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 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.

Support	Resident	Initialization
Basic supervisor	9846	1714
(sum of MAXPROG values) * 8 + 32	( )	
+ #IABUF (defaults to 20) * 8	( )	
+ #XPSSTK (defaults to 20) * 6	( )	
+ #MECBLST (defaults to 20) * 2	( )	
+ 3-bit mode	32	
+ extended address mode	888	
Disk, diskette, or tape		
Disk(ette) basic	2274	4576
+200 per unit	( )	
+ 47 per performance volume	( )	
+128 per I/O task defined	( )	
+ 3-bit mode	10	
+ extended address mode	136	
DISKIOX	5954	210
DMEXTBL	2954	210
IDSK	1316	
4962/4964	1568	
4963/4967/DDSK-30	652	
4963 with fixed-head		526
+178 per unit	( )	
4965/4966	1640	
1024 Byte/Sector Support	470	
1024 Byte/Sector IPL Support	2896	
4968/4969	5280	1704
+130 per unit	( )	
+128 per I/O task defined	( )	
Terminals		
Basic	5670	1776
MFA (feature 1310)		1708
+50 per adapter	( )	
ALPA (feature 1250)		2478
+50 per adapter	( )	
SMIO (feature 5640)		2478
+50 per adapter	( )	
4979/4978/4980	3016	2550
+488 per 4978/4979/4980	( )	
4234/4973/4974/4975/5219/5224/5225/5262	1262	
+550 per 4973/4974	( )	
+562 per 4975	( )	
+562 per 5219	( )	
+562 per 5224	( )	
+562 per 5225/5262	( )	

Figure E-1 (Part 1 of 3). Supervisor Storage Requirements.

Support	Resident	Initialization
4013 type devices	592	182
+458 per device	( )	
Virtual terminals	706	
+470 per terminal	( )	
Series/1 to Series/1	1330	274
+624 per device	( )	
GPIB	810	
+574 per adapter	( )	
Basic for		
any TTY, 2741/PROC, non-3101B, ACCA	768	
+514 if ASCII/EBCDIC conversion	( )	
+514 if mirror image ASCII/EBCDIC conversion	( )	
Teletypewriter	768	
+470 per teletypewriter	( )	
2741/PROC	1572	
+594 per 2741/PROC	( )	
+532 if correspondence code	( )	
+522 if EBCD code	( )	
ACCA ASCII Terminals no MFA	1704	
+530 per terminal	( )	
+ACCA trace facility	1388	
+No ACCA trace facility	8	
3101 (block mode)	2250	
+724 per device	( )	
Spool		
Basic (in supervisor)	2412	
+(498*max active jobs)+32	( )	
(Spool to 4224/4201 printer)		
+(596*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	1326	356
4952/4954/4956	1374	188

Figure E-1 (Part 2 of 3). Supervisor Storage Requirements.

Support	Resident	Initialization
Binary synchronous communications access method		
Any BSC device	4470	578
In addition:		
+ 22 per multiline feature	( )	
+136 per line	( )	
+3-bit mode	6	
+extended address mode	109	
X.21 Facility	2002	
Host Communication Facility	2346	364
Sensor-based input/output		
Basic	1078	180
Analog input	682	
+48 for first AI group	( )	
+16 for each additional group	( )	
Analog output	68	
+16 per AO	( )	
Digital input and output	946	
+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	1374	66
+(32+x(16+n)) per device	( )	
where: x=maximum number of DCBs n=number of residual status bytes transferred		
+EXIO trace	774	
+No EXIO trace	10	
Relocating loader	6224	3642
+3-bit mode	6	
+extended address mode	156	
Floating point		
included	370	
not included	6	
Support of GETEDIT/PUTEDIT	1628	
Queue processing	304	
$DEBUG	676	
SRMGR (extended address mode)	538	
SCBTBL	2084	
Supervisor patch area	256	

Figure E-1 (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 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     SYSPARTS     NUMPART=___,
                       PARTS=(___,___,___,___,___,___,___,___,
                              ___,___,___,___,___,___,___,___,
                              ___,___,___,___,___,___,___,___,
                              ___,___,___,___,___,___,___,___),
                       MAXPROG=(___,___,___,___,___,___,___,___,
                              ___,___,___,___,___,___,___,___,
                              ___,___,___,___,___,___,___,___,
                              ___,___,___,___,___,___,___,___)
```

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."

System definition statements				Required or Optional
Number of partitions, programs and size definitions				
blank	SYSPARTS	NUMPART=	___,	Required
		PARTS=	(___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___),	Optional
		MAXPROG=	(___ ___ ___ ___, ___)	Optional
System buffer sizes definitions				
blank	SYSPARMS	DATEFMT=	_____,	Optional
		IABUF=	_____,	Optional
		INITMOD=	_____,	Optional
		INITPRT=	_____,	Optional
		LOGPART=	_____,	Optional
		MECBLST=	_____,	Optional
		TBPART=	_____,	Optional
		VIRPART=	_____,	Optional
		XPSSTK=	_____	Optional
Base and common memory partition definition				
blank	SYSCOMM	COMMON=	(___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___, ___ ___ ___ ___),	Required

Figure E-2 (Part 1 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
		COMBASE=	(__ __ __ __, __ __ __ __, __ __ __ __, __ __ __ __, __ __ __ __, __ __ __ __, __ __ __ __, __ __ __ __)	Optional
End of system generation area definition				
blank	SYSEND			Required
Tape definition				
blank	TAPE	DEVICE=	**4969,**	Required
		ADDRESS=	_____,	Required
		DENSITY=	_____,	Optional
		LABEL=	_____,	Optional
		ID=	_____,	Required
		TASK=	_____,	Optional
		END=	_____	Optional
blank	TAPE	DEVICE=	**4968,**	Required
		ADDRESS=	_____,	Required
		DENSITY=	_____,	Optional
		LABEL=	_____,	Optional
		ID=	_____,	Required
		TASK=	_____,	Optional
		END=	_____	Optional
Disk(ette) definitions				
blank	DISK	DEVICE=	**4962,**	Required
		ADDRESS=	_____,	Required
		VOLNAME=	(__ __ __ __),	Optional
		TASK=	_____,	Optional
		END=	_____	Optional

Figure E-2 (Part 2 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
blank	DISK	DEVICE=	**4963**,	Required
		ADDRESS=	_____,	Required
		VOLNAME=	(___, ___, ___, ___),	Optional
		TASK=	_____,	Optional
		END=	_____	Optional
blank	DISK	DEVICE=	_____,	Required (4967-2 or 4967-2A or 4967-4 or 4967-4A)
		ADDRESS=	_____,	Required
		VOLNAME=	(___, ___, ___, ___),	Optional
		TASK=	_____,	Optional
		END=	_____	Optional
blank	DISK	DEVICE=	**DDSK30,**	Required
		ADDRESS=	_____,	Required
		VOLNAME=	(___, ___, ___, ___),	Optional
		TASK=	_____,	Optional
		END=	_____	Optional
blank	DISK	DEVICE=	**DDSK60,**	Required
		ADDRESS=	_____,	Required
		VOLNAME=	(___, ___, ___, ___),	Optional
		TASK=	_____,	Optional
		END=	_____	Optional
blank	DISK	DEVICE=	**IDSK,**	Required
		ADDRESS=	_____,	Required
		VOLNAME=	(___, ___, ___, ___),	Optional
		TASK=	_____,	Optional
		END=	_____	Optional

Figure E-2 (Part 3 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Disk(ette) definitions (continued)				
blank	DISK	DEVICE=	**4964,**	Required
		ADDRESS=	_____,	Required
		VOLNAME=	(___, ___, ___, ___),	Optional
		TASK=	_____,	Optional
		END=	_____	Optional
blank	DISK	DEVICE=	**4965,**	Required
		ADDRESS=	_____,	Required
		VOLNAME=	(___, ___, ___, ___),	Optional
		TASK=	_____,	Optional
		END=	_____	Optional
blank	DISK	DEVICE=	**4966,**	Required
		ADDRESS=	_____,	Required
		VOLNAME=	(___, ___, ___, ___),	Optional
		TASK=	_____,	Optional
		END=	_____	Optional
Timer definition				
blank	TIMER	ADDRESS=	_____	Required
Adapter definitions				
label	ADAPTER	ADDRESS=	_____,	Required
		TYPE=	**MFA,**	Required
		DEVICES=	(___, ___, ___, ___),	Required
		END=	_____	Optional
label	ADAPTER	ADDRESS=	_____,	Required
		TYPE=	**ALPA,**	Required
		DEVICES=	(___, ___, ___, ___, (___, ___, ___, ___),	Required
		END=	_____	Optional
label	ADAPTER	ADDRESS=	_____,	Required
		TYPE=	**SMIO,**	Required
		DEVICES=	(___,___,___,___, ___, ___, ___, ___),	Required
		END=	_____	Optional

Figure E-2 (Part 4 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Basic definition for system message logging				
blank	SYSMSG	TERM=	_____,	Optional
		DISK=	_____,	Optional
		CF=	_____	Optional
Terminal definitions				
Virtual Terminals				
label	TERMINAL	DEVICE=	**VIRT,**	Required
		ADDRESS=	_____,	Required-enter label of other virtual terminal
		LINSIZE=	_____,	Optional
		SYNC=	_____,	Optional
		END=	_____	Optional
label	TERMINAL	DEVICE=	**VIRT,**	Required
		ADDRESS=	_____,	Required-enter label of other virtual terminal
		LINSIZE=	_____,	Optional
		SYNC=	_____,	Optional
		END=	_____	Optional
Series/1-Series/1				
label	TERMINAL	DEVICE=	**S1S1,**	Required
		ADDRESS=	_____,	Required
		LINSIZE=	_____,	Optional
		CHKSUM=	_____,	Optional
		END=	_____	Optional

Figure E-2 (Part 5 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Terminal definitions (continued)				
GPIB				
label	TERMINAL	DEVICE=	**GPIB,**	Required
		ADDRESS=	_____,	Required
		CODTYPE=	_____,	Optional
		LINSIZE=	_____,	Optional
		ATTN=	_____,	Optional
		SCREEN=	_____,	Optional
		PART=	_____,	Optional
		SPOOL=	_____,	Optional
		END=	_____	Optional
Processor-to-Processor				
label	TERMINAL	DEVICE=	**PROC,**	Required
		ADDRESS=	_____,	Required (in hex)
		CODTYPE=	_____,	Optional
		LINSIZE=	_____,	Optional
		CRDELAY=	_____,	Optional
		BITRATE=	_____,	Optional
		RANGE=	_____,	Optional
		CR=	_____,	Optional
		LF=	_____,	Optional
		END=	_____	Optional

Figure E-2 (Part 6 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Terminal definitions (continued)				
2741 Terminal				
label	TERMINAL	DEVICE=	**2741,**	Required
		ADDRESS=	_____,	Required
		PAGSIZE=	_____,	Optional
		LINSIZE=	_____,	Optional
		CODTYPE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		LINEDEL=	_____,	Optional
		CHARDEL=	_____,	Optional
		CRDELAY=	_____,	Optional
		ECHO=	_____,	Optional
		BITRATE=	_____,	Optional
		ADAPTER=	_____,	Optional
		CR=	_____,	Optional
		LF=	_____,	Optional
		SCREEN=	_____,	Optional
		PART=	_____,	Optional
		SPOOL=	_____,	Optional
		END=	_____	Optional

Figure E-2 (Part 7 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Terminal definitions (continued)				
4013 Graphics Terminal				
label	TERMINAL	DEVICE=	4013,	Required
		PAGSIZE=	_____,	Optional
		LINSIZE=	_____,	Optional
		CODTYPE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		LINEDEL=	_____,	Optional
		CHARDEL=	_____,	Optional
		CRDELAY=	_____,	Optional
		ECHO=	_____,	Optional
		CR=	_____,	Optional
		LF=	_____,	Optional
		SCREEN=	_____,	Optional
		DI=		
		DO=	_____,	Required (one only)
		PI=		
		PART=	_____,	Optional
		END=	_____	Optional
4224/4201 Printers				
label	TERMINAL	DEVICE=	_____,	Required (4224/4201)
		ADDRESS=	_____,	Required
		PAGSIZE=	_____,	Optional
		LINSIZE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		SPOOL=	_____,	Optional
		ADAPTER=	_____,	Required (if MFA)
		TIMERS=	_____,	Optional
		LMODE=	_____,	Optional
		RANGE=	_____,	Optional
		CODTYPE=	_____,	Optional
		BITRATE=	_____,	Optional
		END=	_____	Optional

Figure E-2 (Part 8 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Terminal definitions (continued)				
4973/4974 Printers				
label	TERMINAL	DEVICE=	_____,	Required (4973/4974)
		ADDRESS=	_____,	Required
		PAGSIZE=	_____,	Optional
		LINSIZE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		SPOOL=	_____,	Optional
		END=	_____	Optional
4975 Printer				
label	TERMINAL	DEVICE=	**4975,**	Required
		ADDRESS=	_____,	Required
		PAGSIZE=	_____,	Optional
		LINSIZE=	_____,	Optional
		CHARSET=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		BITRATE=	_____,	Optional
		LMODE=	_____,	Optional
		ADAPTER=	**MFA,**	Required
		SPOOL=	_____,	Optional
		END=	_____	Optional

Figure E-2 (Part 9 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Terminal definitions (continued)				
4978/4979 Terminals				
label	TERMINAL	DEVICE=	_____,	Required (4978/4979)
		ADDRESS=	_____,	Required
		LINSIZE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		NHIST=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		HDCOPY=	_____,	Optional
		ATTN=	_____,	Optional
		PF1=	_____,	Optional
		SCREEN=	_____,	Optional
		PART=	_____,	Optional
		SPOOL=	_____,	Optional
		END=	_____	Optional

Figure E-2 (Part 10 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Terminal definitions (continued)				
4980 Terminals				
label	TERMINAL	DEVICE=	**4980,**	Required
		ADDRESS=	_____,	Required
		LINSIZE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		NHIST=	_____,	Optional
		BITRATE=	_____,	Required
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		HDCOPY=	_____,	Optional
		ATTN=	_____,	Optional
		PF1=	_____,	Optional
		SCREEN=	_____,	Optional
		PART=	_____,	Optional
		SPOOL=	_____,	Optional
		ADAPTER=	**SMIO,**	Required
		PORT=	_____,	Required (0 or 1)
		SECADDR=	_____,	Required (00 – FE)
		END=	_____	Optional
5219/5224/5225/5262 Terminals				
label	TERMINAL	DEVICE=	_____,	Required (5219/ 5224/5225)
		ADDRESS=	_____,	Required
		PAGSIZE=	_____,	Optional
		LINSIZE=	_____,	Optional
		CHARSET=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		LMODE=	_____,	Optional
		ADAPTER=	**ALPA,**	Optional
		SPOOL=	_____,	Optional
		PORT=	_____,	Required (0 or 1)
		SECADDR=	_____,	Required (00 – 06)
		END=	_____	Optional

Figure E-2 (Part 11 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Terminal definitions (continued)				
ACCA-type Terminals				
label	TERMINAL	DEVICE=	**ACCA,**	Required
		ADDRESS=	_____,	Required
		MODE=	_____,	Required
		PAGSIZE=	_____,	Optional
		CODTYPE=	_____,	Optional
		LINSIZE=	_____,	Optional
		TOPM=	_____,	Optional
		NHIST=	_____,	Optional
		BOTM=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		LINEDEL=	_____,	Optional
		CHARDEL=	_____,	Optional
		CRDELAY=	_____,	Optional
		BITRATE=	_____,	Optional
		RANGE=	_____,	Optional
		LMODE=	_____,	Optional
		ADAPTER=	_____,	Optional
		COD=	_____,	Optional
		CR=	_____,	Optional
		LF=	_____,	Optional
		ATTN=	_____,	Optional
		PF1=	_____,	Optional
		SCREEN=	_____,	Optional
		PART=	_____,	Optional
		TIMERS=	_____,	Optional
		SPOOL=	_____,	Optional
		END=	_____	Optional

Figure E-2 (Part 12 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
Terminal definitions (continued)				
TTY Terminals				
label	TERMINAL	DEVICE=	**TTY,**	Required
		ADDRESS=	_____,	Required
		MODE=	_____,	Optional
		PAGSIZE=	_____,	Optional
		CODTYPE=	**ASCII,**	Required
		LINSIZE=	_____,	Optional
		TOPM=	_____,	Optional
		BOTM=	_____,	Optional
		LEFTM=	_____,	Optional
		RIGHTM=	_____,	Optional
		OVFLINE=	_____,	Optional
		LINEDEL=	_____,	Optional
		CHARDEL=	_____,	Optional
		CRDELAY=	_____,	Optional
		ECHO=	_____,	Optional
		CR=	_____,	Optional
		LF=	_____,	Optional
		ATTN=	_____,	Optional
		PF1=	_____,	Optional
		SCREEN=	_____,	Optional
		PART=	_____,	Optional
		SPOOL=	_____,	Optional
		END=	_____	Optional
Binary Synchronous Line Definition				
label	BSCLINE	ADDRESS=	_____,	Optional
		TYPE=	_____,	Optional
		RETRIES=	_____,	Optional
		MC=	_____,	Optional
		ADAPTER=	_____,	Optional
		POLL=	(__, __, __, __),	Optional
		END=	_____	Optional
Local Communications Controller Attachment				
label	LCC	ADDRESS=	_____,	Required
		END=	_____	Optional

Figure E-2 (Part 13 of 14). Work Sheet 2 - Define System Configuration

System definition statements				Required or Optional
EXIO Device Interface Definition				
blank	EXIODEV	ADDRESS=	_____,	Required
		MAXDCB=	_____,	Optional
		RSB=	_____,	Optional
		EXAR=	_____,	Optional
		DCBRSB=	_____,	Optional
		END=	_____	Optional
Host Communication Support				
blank	HOSTCOMM	DEVICE=	**BSCA,**	Required
		ADDRESS=	_____	Required
Sensorio Device Definition				
blank	SENSORIO	DEVICE=	_____,	Required (IDIO/4952)
		ADDRESS=	_____,	Required
		AI=	_____,	Optional
		AO=	_____,	Optional
		DI=	_____,	Optional
		DO=	_____,	Optional
		PI=	_____,	Optional
		AITYPE=	_____,	Optional
		LEVEL=	_____,	Optional
		END=	_____	Optional
System Common				
	$SYSCOM	CSECT		
		QCB		
		QCB		
		ECB		
		ECB		
		ENTRY	$EDXPTCH	
	$EDXPTCH	DATA	128F'0'	System Patch Area

Figure E-2 (Part 14 of 14). Work Sheet 2 - Define System Configuration

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 4-1 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.

Sample Work Sheet 3

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
	PART 1		
	Supervisor support	-	must be first and in partition 1
	VOLUME XS6006		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 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
	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
	Host communications support	-	must be in partition 1
	INCLUDE TPCOM	*14*	Host communication support
	Translation tables	-	must be in partition 1

Figure E-3 (Part 1 of 4). Work Sheet 3 - Installation and System Generation

Column One	Supervisor Object Modules	Notes	Purpose of Module
	INCLUDE TRASCII	*10*	1310,2095/2096,7850 ACCA/TTY translation
	INCLUDE TREBASC	*10*	1610, 2091/2092 ACCA translation
	INCLUDE TREBCD	*11*	2741, PROC EBCD translation
	INCLUDE TRCRSP	*11*	2741 correspondence translation
	System initialization	-	must be in partition 1
	INCLUDE $PROG1	*22*	User module included in nucleus gen
	INCLUDE IO1024	*21*	1024 IPL support
	SNA EDL stub module		
	INCLUDE $SNASTUB	*33*	SNA EDL instruction stub module
	System support - initialization	-	must be in partition 1
	INCLUDE EDXINIT	*24*	Supervisor initialization
	PART 2		
	INCLUDE SUPVIO	*28*	Make supervisor and common area static
	INCLUDE EDXALU	*30*	EDL instruction emulator
	INCLUDE DLCROUTE	*34*	Data link control router
	INCLUDE CDXQAPND	*34*	Data link control router sub module
	INCLUDE CDXQPULL	*34*	Data link control router sub module
	INCLUDE CDXQWAIT	*34*	Data link control router sub module
	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
	INCLUDE DISKIO	*3*	Basic disk(ette) support
	INCLUDE DISKIOX	*31*	Dynamic data set extent support - optional
	INCLUDE D49624	*3*	4962/4964 disk(ette) support
	INCLUDE D4963A	*3*	4963/4967/DDSK disk support
	INCLUDE D4966A	*3*	4965/4966 diskette support
	INCLUDE DIDSKA	*3*	IDSK disk(ette) support
	INCLUDE D1024	*3,21*	1024 bytes/sector diskette support

Figure E-3 (Part 2 of 4). Work Sheet 3 - Installation and System Generation

Column One	Supervisor Object Modules	Notes	Purpose of Module
	INCLUDE D4969A	*3*	Basic tape support
	Terminals	-	Must be grouped together May be included in partition 1 to 8
	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 IOS316X	*33*	ACCA 3161B/3163B/3164B support (3151)
	INCLUDE IOS4979	*3*	4978/4979/4980 display support
	INCLUDE IOS4974	*3*	4234/4973/4974/4975/5219/5224 /5225/5262 support
	INCLUDE IOS4224	*3*	4201/4202/4224 TERMCTRL support
	INCLUDE IOS4975A	*3*	ACCA 4975-01A/4201/4202/4224 support
	INCLUDE IOSACCA	*4*	ACCA device handler
	INCLUDE ACCATRC	*3*	ACCA trace option
	INCLUDE IOSHARE	*3*	ACCA line sharing 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 included in partition 1 to 8
	INCLUDE EDXFLOAT	*3*	Floating point arithmetic
	Queue I/O	-	May be included in partition 1 to 8
	INCLUDE QUEUEIO	*19*	Queue processing support
	Bisync communications	-	Must be grouped together May be included in partition 1 to 8
	INCLUDE BSCAM	*13*	Bisync communication

Figure E-3 (Part 3 of 4). Work Sheet 3 - Installation and System Generation

Column One	Supervisor Object Modules	Notes	Purpose of Module
	INCLUDE BSCX21	*27*	Bisync communication support for X.21
	Sensor input/output	-	Must be grouped together May be included in partition 1 to 8
	INCLUDE SBCOM	*15*	Basic sensor I/O support
	INCLUDE SBAI	*3*	Analog input support
	INCLUDE SBAO	*3*	Analog output support
	INCLUDE SBDIDO	*3*	Digital I/O support
	INCLUDE SBPI	*3*	Process interrupt support
	LCC communications	-	May be included in partition 1 to 8
	INCLUDE LCCAM	*3*	Local Communications Controller support
	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

Figure E-3 (Part 4 of 4). Work Sheet 3 - Installation and System Generation

Column One	Supervisor Object Modules	Notes	Purpose of Module
	Insert user initialization modules here		

Figure E-4. Work Sheet 3 - Installation and System Generation (continued)

Column One	Supervisor Object Modules	Notes	Purpose of Module
	Supervisor code moved out of partition 1 must be moved here		
	Location of supervisor		
	LINK $EDXNUCT,EDX002		REPLACE END

Figure E-5. Work Sheet 3 - Installation and System Generation (continued)

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:

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 E-28 is a list of object modules that must remain in partition 1 and their approximate sizes (in bytes). Under "Partition Any" on page E-30 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).

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 (required)	1382
EDXSVCX (required)	2892
$DBUGNUC	676
EDXSTART (required)	2288
SWAITM	954
EDXTIMER	1326
EDXTIMR2	1374
IOSEXIO	1554
EXIOTRC	820
NOEXIOTR	10
RLOADER (LOADBUFR)	6266
LDRCHK (extended address mode)	164
NOLDRCHK (3-bit mode)	6
STORMGR	1644
NOSTRMGR	26
IAMQCB	26
PWRAM80	366
TPCOM (TPCOM1)	2346
TRASCII	514
TREBASC	514
TREBCD	514
TRCRSP	514
IO1024	2866
SYSLOG (required)	266
CIRCBUFF (required)	252
XPSTABLE (required)	482
SRCHK (extended address mode)	1042
NOSRCHK (3-bit mode)	20
SYSLOGIA	144
User initialization modules	

Figure E-6. Supervisor Modules Required in Partition 1

Overlay Initialization Modules: 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 initialization modules if you include the OPTION NOVERLAY statement. If $XPSLINK includes them as resident initialization modules, 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 E-6 on page E-28.

Note: If you define OVERLAY, the largest size you will need is 2560 bytes.

MODULE NAME	SIZE (IN BYTES)
$BSCRAM	770
BSCINIT	648
CLOKINIT	188
DSKINIT1	1760
DSKINIT2	1194
DSKXINIT	200
EXIOINIT	66
INITMFA	1422
INITADAP	2480
INIT4013	182
INIT4978	2396
INIT4980	2550
LCCINT	752
LCPOV1	1956
LCPOV2	1560
LOADINIT	1568
LOADINT2	2074
LOGLOAD	858
OPENADAP	1812
OPENMFA	1708
OPEN4980	1852
RW4963ID	526
SBIOINIT	180
SRINIT1	1860
SRINIT2	1794
SROPEN	1698
S1S1INIT	274
STORINIT	590
TAPEINIT	1704
TERMINIT	1988
TERMOPEN	1844
TIMRINIT	188
TPINIT	364

Resident Initialization Modules: The following modules are resident initialization modules. You cannot include them in an overlay area.

MODULE NAME	SIZE (IN BYTES)
$OVLMGR0	132
CMDINIT	122
EDXINIT	10
EDXINIT1	888
INITMODS	94
LCCPTC	514
SEGINIT	506
XPSINIT	2154

The following supervisor modules are required in partitions 2 – 8 with other supervisor support included in these partitions.

MODULE NAME	SIZE (IN BYTES)
XPSSYS	134
SRCHK (extended address mode)	1048
NOSRCHK (3-bit)	20

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 E-6 on page E-28.

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 E-6 on page E-28.

Module Name	Size (in Bytes)	Root Segment
DIDSKA	960	356
DISKIO (DISKTBL)	1530	778
DISKIOX	5954	0
DMEXTBL (DISKIOX default table size)		
D1024	312	158
D4950	748	304
D49624	1554	22
D4963A	620	18
D4966A	1004	644
D4969A	4146	1280
DSKCHK (extended address mode)	136	0
NODSKCHK (3-bit mode)	10	0
EDXALU	2372	12
EDXTERMQ	1236	0
IOSTABLE	58	0
EDXTIO	5012	780
NOTIO (same part. as EDXALU)	188	0
MINMSG	244	0
FULLMSG	1360	0
IOS3101	2274	2
IOS316X	2750	0
IOS4224	1708	0
NO4224	34	0
IOS4974	1214	86
IOS4975A	1412	902
IOS4979	2782	292
IOSHARE	682	
IOSTERM	746	0
IOSACCA	1592	414
ACCATRC	714	678
NOACCATR	8	0
IOSTTY	172	516
IOS2741	878	694
IOS4013	282	310
IOSGPIB	676	134
IOSS1S1	986	344
IOSVIRT	740	2
IOSPOOL	2320	194
EBFLCVT	1576	0
SRMGR	266	272
SCBTBL	2052	0
SMLLOAD	738	0

Figure E-7 (Part 1 of 2). Supervisor Modules Not Required in Partition 1

Module Name	Size (in Bytes)	Root Segment
EDXFLOAT	362	8
NOFLOAT (same part. as EDXALU)	6	0
QUEUEIO	312	0
BSCAM	4400	86
BSCX21	456	96
BSCCHK (extended address mode)	104	0
NOBSCCHK (3-bit mode)	6	0
X21CMD	1430	0
IOLOADER	982	0
SBCOM	84	12
SBAI	632	50
SBAO	68	0
SBDIDO	778	168
SBPI	2	166
LCCAM	1834	132

Figure E-7 (Part 2 of 2). Supervisor Modules Not Required in Partition 1

PART B

Program Products

If you are including other program products in your system, they will require supervisor storage in partition one. Read down the following list and cross out the products you are not including in your system. Refer to the product directory for each product you are including for that product's storage requirements.

- 5230 Data Collection
- System/370 Channel Attach
- Multiple Terminal Manager (MTM)
- Support of 1024 bytes per 2200 sector diskette
- Indexed Access Method (IAM)
- Communications Facility (CF)
- Transaction Processing System (TPS).

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	2048	
$BSCUT1	6912	
$BSCUT2	20736	
$COMPRES	12544	
$COPY	14080	
$COPYUT1	18944	
$CPUMON	5888	
$CPUPRT	4352	
$DASDI	2816	
$DEBUG	13312	
$DICOMP	13056	
$DIINTR	9984	
$DIRECT	13312	
$DISKUT1	19456	
$DISKUT2	23296	(+1280 if printing error log)
$DISKUT3	6400	(used in loading)
$DIUTIL	11520	
$DSKPRT1	12032	
$DSKMON	15104	(loads $DSKMON2 and $DSKDDB)
$DSKPRT2	16384	
$DUMP	25600	
$EDIT1	10496	
$EDIT1N	14080	
$EDXASM	20736	(+5632 if assembling TERMINAL statements)
$EDXLINK	23296	
$EDXLIST	10496	
$FONT	18688	
$FSEDIT	24320	
$GPIBUT1	12544	
$HCFUT1	2816	
$HXUT1	18944	
$IAMUT1	22016	(used for Indexed Access Method Versions 1 and 2)
$IMAGE	36608	
$INITDSK	25856	
$INSTAL	17152	
$IOTEST	13824	
$JOBQUT	6656	
$JOBUTIL	1280	

Utility	Size	
$LCCTRCE	4864	
$LCCUT1	9472	
$LINSET	8192	
$LINTRC	16896	
$LINUT1	54784	
$LOG	8192	
$MEMDISK	11776	(refer to the Operator Commands and Utilities Reference for additional storage requirements)
$MOVEVOL	20736	
$MSGUT1	15360	
$PCUTIL	18944	
$PDS	2048	
$PFMAP	512	
$PREFIND	7168	
$PRT2780	2560	
$PRT3780	2816	
$RJE2780	11776	
$RJE3780	12544	
$RMU	10240	
$SPLUT1	8448	
$S1PPRG	8704	
$S1PPRGR	5120	
$S1PSYS	15616	
$S1PSYSR	20992	
$S1S1UT1	9984	
$STGUT1	37888	
$SUBMIT	7936	(if $JOBQ must be in same partition, add 4096)
$TAPEUT1	7168	(refer to the Operator Commands and Utilities Reference for storage needed for commands)
$TERMUT1	13312	
$TERMUT2	19456	
$TERMUT3	1024	
$TRACEIO	14080	
$TRANS	13312	(for transmitting end)
$TRANS	10752	(for receiving end)
$TRAP	16640	(and 512 in partition 1)
$UPDATE	9984	
$UPDATEH	7680	
$XPSLINK	24832	

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	= (324)
11.	$DISKUT3 data management utility	= (4384)
12.	$LOADER transient program loader	= (6280)
13.	$UPCASE	= (150)
14.	$4975	= (462)
	TOTAL	= ( )

The following notes supply additional information on how to determine these storage items:

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	alphanumeric	202
GETEDIT	float.pt. F or E	96
GETEDIT	integer	90
PUTEDIT	alphanumeric	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
```

```
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	3020
$PACK	146
$UNPACK	98
$IMDTYPE	218
$IMGEN (includes $IMDEFN, $IMPROT, $IMDATA)	4568
$IMGEN31	2148
$IMGEN49	1200
$IMGEN3X	4224
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	(Bytes) SIZE	X Occurrence	
$$RETURN (entry point RETURN)	42	X _____	= _____
$$SVC (entry point SVC)	36	X _____	= _____
$EDXATSR (entry points) - SETBUSY, SUPEXIT	74	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 $DISKUT3 in a program, it requires an additional 4600 bytes of storage.
12. The transient program loader ($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.

Appendix F. Setting Up the 4201, 4202, and 4224 Printers

Set-Up Operations

The following section shows you operations you need to perform for setting up the 4201, 4202, and 4224 printers.

Setting Up the 4201

Before you set up the 4201 printer, you must install the Serial Interface Card, #PN6493187, by following the directions in the Proprinter Serial Interface Card Setup Instructions (#PN6493226).

Before you set any switches, set the printer power-off switch to OFF. Then set switches 1 through 7 on the parallel attachment card and switches A1 through A6 and B1 through B4 on the serial interface card as follows:

- Set switch 1 (Beeper) the way you want it.
- Set switch 2 (Slashed Zeros) the way you want it.
- Set switch 3 (Automatic Line Feed) to OFF.
- Set switch 4 (Form Length) to match the physical forms.
- Set switch 5 (Character Set) to ON to select character set PC2.
- Set switch 6 (Automatic Carriage Return) to OFF.
- Set switch 7 (Reserved) to OFF.
- Set switches A1, A2, A3 to the bit rate you specified on the TERMINAL statement. If your 4201/4202 is connected to the auxiliary port of a 3151, 3161, 3163 or 3164 terminal for line sharing, set switches A1, A2, A3 to the bit rate specified in the software switches for the auxiliary port of the terminal. The following chart shows the software switch settings and bit rates.

A1	A2	A3	Speed
off	off	off	9600 bps
off	off	on	4800 bps
off	on	off	2400 bps
off	on	on	1200 bps
on	off	off	600 bps
on	off	on	300 bps
on	on	off	150 bps
on	on	on	19200 bps

Figure F-1. Software Switches and Bit Rates

- Set switches A4, A5 to OFF to select no parity.
- Set switch A6 to OFF to select XON/XOFF protocol.
- Set switch B1 to OFF to select 8 bits per character.
- Set switch B2 to ON to select 2 stop bits for each frame.
- Set switch B3 to ON for the print test; otherwise, set it to OFF.
- Set switch B4 to OFF (always OFF).

Note: If you IPL from the starter system, leave your 4201 switched off until you perform a customized system generation that includes the 4201. Unpredictable results can result if the starter system initially assumes another device at that address.

If you are using asynchronous line sharing and the 4201 printer is connected to the Personal Computer (PC) (with the Intelligent Work Station installed on the PC), do not use the Serial Interface Card on the 4201 printer. The PC is connected directly to the parallel attachment card of the 4201 printer.

Setting Up the 4202

Before you set up the 4202 printer, you must install the Serial Interface Card, #PN6493187, by following the directions in the Proprinter Serial Interface Card Setup Instructions (#PN6493226).

Before you set any switches, set the printer power-off switch to OFF. Then set switches 1 through 7 on the parallel attachment card and switches A1 through A6 and B1 through B4 on the serial interface card. Set switch 1 (line size) ON to select a 13.6 inch line. Set switch 1 OFF to select an 8 inch line. Set all other switches, 2 through 7 and A1 through B4, to select functions as indicated in "Setting Up the 4201" on page F-1.

Setting Up the 4224

As part of the setup operation for the 4224, you should run several offline tests. These tests set the internal state of the printer so the Series/1 can communicate with it. Once you run the tests, the printer remembers the state even if you switch the power off.

The following are tests that you must run before you can begin printer operations. Refer to the *IBM 4224 Printer Operating Instructions,* GC31-2545, or the *IBM 4224 Printer Setup Instructions,* GC31-3607, for an explanation of these tests and how to run them.

- Run test 300 to align the forms.
- Run test 302 to set the power-on defaults as follows:
  1. Set the characters per inch (CPI) in conjunction with the maximum print position (MPP). The system forces CPI to 10 at IPL time, but you can alter CPI with the TERMCTRL statements or with the $TERMUT1 utility.
  2. Set the lines per inch (LPI) in conjunction with the maximum print length (MPL). The system forces LPI to 6 at IPL time, but you can alter LPI with the TERMCTRL statements or with the $TERMUT1 utility.

3. Set MPP in conjunction with CPI to obtain the maximum print line, in inches, for the printer forms you are using. For example, if the print line is 13.2 inches and CPI is set to 10, you must set MPP to 132.
4. Set MPL in conjunction with LPI to obtain the maximum print length, in inches, for the printer forms you are using. For example, if the forms are 11 inches in length and LPI is set to 6, you must set the MPL to 66.
5. You can set your print quality the way you want it. The system forces it to "data processing" at IPL time, but you can change print quality with the TERMCTRL statements or with the $TERMUT1 utility.

- Run test 303 to select PC character set 2 for English or to select one of the character sets for languages other than English.
- Run test 305 to set the bit rate to match what you specified on the TERMINAL statement.
- Run test 312 to select form feed suppression at top of forms if you have MODE=PAGE or MODE=VERPAGE specified on the TERMINAL statement. This option helps you save paper.
- Run test 315 to set the electrical connection (either RS-422-A or RS-232-C) to match what you specified on the TERMINAL statement.
- Run test 316 to enable status reporting. This allows the printer hardware errors to be sent to the Series/1 and logged to the system log data set.
- Run test 317 to select XON/XOFF pacing protocol.
- Run test 318 to select no parity.

Note: If you IPL from the starter system, leave your 4224 switched off until you perform a customized system generation which includes the 4224.

Cabling for the 4201, 4202, and 4224 Printers

Use attachment cable feature code 2056 (PN1632211) to connect the Series/1 #1310 (RS-232-C port) or the #2095/2096 attachment card to a 25-pin male-to-male Electronic Industries Association (EIA) extender cable. Then connect the extender cable to the serial attachment card of the 4201, 4202, or 4224 printer. For remote attachment, use the attachment cable feature 2057 (PN1632208).

Use attachment cable feature code 5770 (PN6844552) to connect the Series/1 #1310 (RS-422-A port) to the serial attachment card of the 4224 printer.

Appendix G. Alternative Installation Instructions

This appendix contains alternative instructions for the following steps:

- Copy the Starter Supervisor, Basic Utilities, and Support Modules
- Copy the Production Utilities and System Support Modules
- Copy the Program Preparation and Session Manager Modules.

These instructions were formerly in Chapter 3. A more automated procedure using $INSTAL is available. See "Step 4 - Copy the Starter Supervisor, Basic Utilities, and Support Modules Using $INSTAL" on page 3-13 for information on using the $INSTAL utility.

Note: No history file information will be kept if these alternative installation instructions are used.

Step 1 - Copy the Starter Supervisor, Basic Utilities, and Support Modules

Use the $COPYUT1 utility to copy the starter system $EDXNUC on XS6001 to $EDXNUC on EDX002, along with the basic utilities and support modules.

1(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       nnnP, LP= nnnn, PART=x
$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 XS6001, OK (Y/N)?  Y
THE DEFINED TARGET VOLUME IS XS6001, OK (Y/N)?  N EDX002
MEMBER WILL BE COPIED FROM XS6001 TO EDX002, OK (Y/N)?  Y

COMMAND (?):  ROLLON
```

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

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

1(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 XS6001 to EDX002. After all the modules on XS6001 are copied to EDX002, $COPYUT1 displays the COMMAND (?): prompt.

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

```
COMMAND (?): EN

$COPYUT1 ENDED
```

1(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       nnnP, LP= nnnn, PART=x

$INITDSK - DISK INITIALIZATION UTILITY

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

COMMAND (?): EN
$INITDSK ENDED
```

Set the IPL Source switch to IPL from the disk unit. Press the LOAD button on the Series/1 console.

Step 2 - Copy the Production Utilities and System Support Modules

Use the $COPYUT1 utility to copy the production utilities from XS6002, XS6003, and XS6004, and the system support modules from XS6005.

2(a). Insert diskette **XS6002** into the diskette unit. If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

For the 4966:

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

```
> $VARYON 22,1
XS6002 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XS6002 ONLINE
```

2(b). Load the utility $COPYUT1 and copy the contents of XS6002.

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

```
> $L $COPYUT1,,*
LOADING $COPYUT1       nnnP, LP= nnnn, PART=x
$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 XS6002
THE DEFINED TARGET VOLUME IS EDX002, OK (Y/N)?  Y
MEMBER WILL BE COPIED FROM XS6002 TO EDX002, OK (Y/N)?  Y
COMMAND (?):  ROLLON

COMMAND (?):  CALL

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

$COPYUT1 displays a list of the modules being copied from XS6002 to EDX002. After all the modules on XS6002 are copied to EDX002, $COPYUT1 displays the COMMAND (?): prompt.

2(c). Insert diskette **XS6003** into the diskette unit. If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

For the 4966:

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

```
> $VARYON 22,1
XS6003 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XS6003 ONLINE
```

2(d). Change the volume and copy the contents of **XS6003**. Respond to the $COPYUT1 prompts as shown.

```
COMMAND (?): CV

THE DEFINED SOURCE VOLUME IS XS6002, OK (Y/N)? N XS6003
THE DEFINED TARGET VOLUME IS EDX002, OK (Y/N)? Y
MEMBER WILL BE COPIED FROM XS6003 TO EDX002, OK (Y/N)? Y
COMMAND (?): CALL

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

$COPYUT1 displays a list of the modules being copied from XS6003 to EDX002. After all the modules on XS6003 are copied to EDX002, $COPYUT1 displays the COMMAND (?): prompt.

2(e). Replace diskette **XS6003** with diskette **XS6004** in the diskette unit.

If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

For the 4966:

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

```
> $VARYON 22,1
XS6004 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XS6004 ONLINE
```

2(f). Change the volume and copy the contents of **XS6004**. Respond to the $COPYUT1 prompts as shown.

```
COMMAND (?):  CV

THE DEFINED SOURCE VOLUME IS XS6003, OK (Y/N)?  N XS6004
THE DEFINED TARGET VOLUME IS EDX002, OK (Y/N)?  Y
MEMBER WILL BE COPIED FROM XS6004 TO EDX002, OK (Y/N)?  Y
COMMAND (?):  CALL

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

$COPYUT1 displays a list of the modules being copied from XS6004 to EDX002. After the modules from XS6004 are copied to EDX002, $COPYUT1 displays the COMMAND (?): prompt.

2(g). Replace diskette **XS6004** with diskette **XS6005** in the diskette unit.

If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

For the 4966:

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

```
> $VARYON 22,1
XS6005 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XS6005 ONLINE
```

2(h). Change the volume and copy the contents of **XS6005**. Respond to the $COPYUT1 prompts as shown.

```
COMMAND (?):  CV

THE DEFINED SOURCE VOLUME IS XS6004, OK (Y/N)?  N XS6005
THE DEFINED TARGET VOLUME IS EDX002, OK (Y/N)?  N ASMLIB
MEMBER WILL BE COPIED FROM XS6005 TO ASMLIB, OK (Y/N)?  Y
COMMAND (?):  CALL

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

$COPYUT1 displays a list of the modules being copied from XS6005 to ASMLIB. After the modules from XS6005 are copied to ASMLIB, $COPYUT1 displays the COMMAND (?): prompt.

2(i) Remove diskette **XS6005** from the diskette unit.

- If you are going to create a tailored operating system, develop application programs on your Series/1, or use the session manager, you need to copy the Program Preparation modules. Proceed to Step 3.
- If not, you can end $COPYUT1 as shown, and proceed to step 8.

```
COMMAND (?): EN

$COPYUT1 ENDED
```

Step 3 - Copy the Program Preparation and Session Manager Modules (5719-XX7)

This step is required only if you are installing 5719-XX7 to develop application programs or to use the utilities and the session manager on your Series/1.

Note: Either the Program Preparation Facilities (5719-XX7) 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 XX7001, XX7002, and XX7003 to ASMLIB, and the program preparation utilities and the session manager from XX7004 to EDX002, perform the steps that follow.

3(a). Insert diskette **XX7001** into the diskette unit.

If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

For the 4966:

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

```
> $VARYON 22,1
XX7001 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XX7001 ONLINE
```

3(b). Change volumes and copy the contents of **XX7001**. Respond to the $COPYUT1 prompts as shown.

```
COMMAND (?):  CV

THE DEFINED SOURCE VOLUME IS XS6005, OK (Y/N)?    XX7001
THE DEFINED TARGET VOLUME IS ASMLIB, OK (Y/N)?
MEMBER WILL BE COPIED FROM XX7001 TO ASMLIB, OK (Y/N)?  Y
COMMAND (?):  CALL

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

$COPYUT1 displays a list of the $EDXASM modules being copied from XX7001 to ASMLIB. After all the modules are copied from XX7001, $COPYUT1 displays the COMMAND (?): prompt.

3(c). Replace diskette **XX7001** with diskette **XX7002** in the diskette unit.

If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

For the 4966:

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

```
> $VARYON 22,1
XX7002 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XX7002 ONLINE
```

3(d). Change the volume and copy the contents of **XX7002**. Respond to the $COPYUT1 prompts as shown.

```
COMMAND (?): CV

THE DEFINED SOURCE VOLUME IS XX7001, OK (Y/N)? N XX7002
THE DEFINED TARGET VOLUME IS ASMLIB, OK (Y/N)? Y
MEMBER WILL BE COPIED FROM XX7002 TO ASMLIB, OK (Y/N)? Y
COMMAND (?): CALL

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

$COPYUT1 displays a list of the modules being copied from XX7002 to ASMLIB. After all the modules are copied from XX7002 to ASMLIB, $COPYUT1 displays the COMMAND (?): prompt.

3(e). Replace diskette **XX7002** with diskette **XX7003** in the diskette unit.

If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

For the 4966:

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

```
> $VARYON 22,1
XX7003 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XX7003 ONLINE
```

3(f). Change the volume and copy the contents of **XX7003**. Respond to the $COPYUT1 prompts as shown.

```
COMMAND (?): CV

THE DEFINED SOURCE VOLUME IS XX7002, OK (Y/N)? N XX7003
THE DEFINED TARGET VOLUME IS ASMLIB, OK (Y/N)? Y
MEMBER WILL BE COPIED FROM XX7003 TO ASMLIB, OK (Y/N)? Y
COMMAND (?): CALL

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

$COPYUT1 displays a list of the modules being copied from XX7003 to ASMLIB. These modules are the copy-code modules required for certain applications. After all the modules are copied from XX7003 to ASMLIB, $COPYUT1 displays the COMMAND (?): prompt.

3(g). Replace diskette **XX7003** with diskette **XX7004** in the diskette unit.

If you are using a 4966 Diskette Magazine Unit, insert the diskette in slot 1 and vary the diskette online. You must also vary on an IDSK diskette unit.

For the 4966:

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

```
> $VARYON 22,1
XX7004 ONLINE ON SLOT 1
```

For IDSK:

```
> $VARYON 61
XX7004 ONLINE
```

3(h). Change the volume and copy the contents of **XX7004**. Respond to the $COPYUT1 prompts as shown.

```
COMMAND (?): CV

THE DEFINED SOURCE VOLUME IS XX7003, OK (Y/N)? N XX7004
THE DEFINED TARGET VOLUME IS ASMLIB, OK (Y/N)? N EDX002
MEMBER WILL BE COPIED FROM XX7004 TO EDX002, OK (Y/N)? Y
COMMAND (?): CALL

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

$COPYUT1 displays a list of the modules being copied from XX7004 to EDX002. These modules are the program preparation utilities and the session manager. After all the modules are copied from XX7004 to EDX002, $COPYUT1 displays the COMMAND (?): prompt.

3(i). Remove diskette **XX7004** from the diskette unit.

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

```
COMMAND (?): EN

$COPYUT1 ENDED
```

- 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-LMA program directory.
- See "Step 8 - Exercise the Utilities and Program Preparation Facilities" on page 3-21 for the remaining information on installation.

Index

Special Characters

A

B

C

D

J

L

M

N

O

P

Q

R

S

T

U

V

W

X

Numerics

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 postage-paid order form as indicated, seal with tape, and mail.

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.
Basic Books:		
Set of the following eight books. (For individual copies, order by book number.)	SBOF-0255	
Advanced Program-to-Program Communication Programming Guide and Reference	SC34-0960	
Communications Guide	SC34-0935	
Installation and System Generation Guide	SC34-0936	
Language Reference	SC34-0937	
Library Guide and Common Index	SC34-0938	
Messages and Codes	SC34-0939	
Operator Commands and Utilities Reference	SC34-0940	
Problem Determination Guide	SC34-0941	
Additional books and reference aids:		
Set of the following three books and reference aids. (For individual copies, order by number.)	SBOF-0254	
Customization Guide	SC34-0942	
Event Driven Executive Language Programming Guide	SC34-0943	
Operation Guide	SC34-0944	
Language Reference Summary	SX34-0199	
Operator Commands and Utilities Reference Summary	SX34-0198	
Conversion Charts Card	SX34-0163	
Reference Aids Storage Envelope	SX34-0141	
Set of three reference aids with storage envelope. (One set is included with order number **SBOF-0254**.)	SBOF-0253	
Binders:		
Easel binder with 1 inch rings	SR30-0324	
Easel binder with 2 inch rings	SR30-0327	
Standard binder with 1 inch rings	SR30-0329	
Standard binder with 1 1/2 inch rings	SR30-0330	
Standard 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®

IBM Series/1 Event Driven Executive
Installation and System Generation Guide
Order No. SC34-0936-0

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.)

SC34-0936-0
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
5414 (Internal Zip)
P.O. Box 1328
Boca Raton, Florida 33429-9960

Fold and tape | Please Do Not Staple | Fold and tape

IBM®

Program Number
5719-XS6, 5719-XX7

File Number
S1-34

SC34-0936-0

Printed in U.S.A.