IBM

Series/1

SC34-0591-0

Event Driven Executive Extended Address Mode and Performance Analyzer User Guide

Version 5.0

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

IBM

Series/1

SC34-0591-0

Event Driven Executive Extended Address Mode and Performance Analyzer User Guide

Version 5.0

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

First Edition (December 1984)

Use this publication only for the purposes stated in the following 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, P. O. Box 1328, Boca Raton, Florida 33432. 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.

About This Book

This book describes the Event Driven Executive Extended Address Mode support and the EDX Performance Analyzer. It contains information on installation and system generation, problem determination, customization, error messages and return codes, how to load and use the Performance Analyzer, and how to improve system performance.

Audience

This book is intended for users of the 4956 Model E or the 4956-60E processors with the Extended Address Mode support or for ***anyone*** who wants to monitor and improve his system's performance with the EDX Performance Analyzer. Readers should have knowledge about system programming and the optimum performance of an operating system.

How This Book Is Organized

The book is divided into two parts. Part 1 contains five chapters relating to the Extended Address Mode. Part 2 contains four chapters relating to the EDX Performance Analyzer.

Part 1. Extended Address Mode

- *Chapter 1. Planning for System Generation* contains the information you need before installing and generating your system.
- *Chapter 2. Generate a Tailored Operating System* describes how to generate a tailored operating system with the Extended Address Mode support.

How This Book Is Organized *(continued)*

- *Chapter 3. Partition Status and Problem Determination* describes the functions available for determining problems that occur while using the Extended Address Mode support.
- *Chapter 4. Supervisor Module Names (CSECTS)* lists the names of the modules you must include in your supervisor in order to use the EDX Extended Address Mode support.
- *Chapter 5. Customizing Your System* explains ways to increase the mapped area within a partition.

Part 2. EDX Performance Analyzer

- *Chapter 6. System Analyzer* provides step-by-step procedures for monitoring your system's performance and generating a report.
- *Chapter 7. Program Analyzer* provides step-by-step procedures for monitoring program performance and generating a report.
- *Chapter 8. Improving System Performance* provides information on improving the performance of your system and monitoring the changes you make to your system.
- *Chapter 9. Performance Analyzer Error Messages* provides an alphabetical list of error messages for both the System Analyzer and the Program Analyzer.

Aids to Using This Book

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

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

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

A Guide to the Library

Refer to the *Library Guide and Common Index*, SC34-0645 for information on the design and structure of the Event Driven Executive, Version 5 library, for a bibliography of related publications, and for an index to the entire library.

Contacting IBM about Problems

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

Contacting IBM about Problems *(continued)*

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

Contents

Contents

Figures

Part 1. Extended Address Mode

Part 1 contains five chapters about the EDX Extended Address Mode support. Only the IBM Series/1 4956 Model E and the 4956-60E processors support this extended address 4-bit architecture (16 partitions).

Notes

Chapter 1. Planning for System Generation

This chapter describes the requirements for using the EDX Extended Address Mode. The Extended Address Mode provides up to sixteen partitions of 64K bytes each; that is, 1024K bytes of mapped storage (addressable) with an additional 1024K bytes of unmapped storage. You can perform I/O operations into any of the addressable storage, but you can map only 512K bytes of addressable storage for I/O at any one time. You must install Version 5 according to the instructions in the *Installation and System Generation Guide* and Chapter 2 of this manual.

Application Programs

The following is a list of considerations for application programs for Extended Address Mode:

- If you use the hardware OIO instruction for I/O, you must direct the I/O to a static area or you will get unpredictable results.
- All DCBs must reside in partition 1 in a static area.
- You cannot use $TCBO as an Event Control Block (ECB).
- Communications applications will run faster in static partitions.

Assigning Static and Dynamic Partitions

In static partitions, the system maps all user and common areas for I/O 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 16 must be dynamic, and partitions 2 through 8 can be either static or dynamic. (See "Step 5 - Edit $SRPROF - IPL Configuration Profile Data Set" on page UG-11 for information on defining partitions with the PARTS= operand.)

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 terminates the programs then issues an error message to $SYSLOG (depending on what you code for the WAITIOSR operand in the $SRPROF data set).

Note: See Chapter 5, "Customizing Your System" on page UG-27 for information on changing the mapped area in each partition.

You can change the mixture of static and dynamic partitions by editing the IPL configuration profile data set, $SRPROF and performing an initial program load (IPL) of the system. Then load $STGUT1 to see how the system is using the dynamic segmentation registers (see "Monitor the Status of System Control Blocks" on page UG-18).

You can assign 7 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 terminate them (depending on what you specify in $SRPROF). 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 terminates the program and issues an error message to $SYSLOG.

If some of your programs perform loads with PART=ANY (see "Coding the LOAD Instruction" on page UG-5 for an example of the LOAD instruction) you can load these programs into static partitions only by taking the default of "S" for the first operand on the LOADER statement in the IPL configuration profile data set, $SRPROF. However, then your static partitions will fill up with programs that do not need to run in static partitions. Change the loads with PART=ANY to PART=STATIC for programs you want to run in static partitions and PART=DYNAMIC for programs you want to run in dynamic partitions.

If you do ***not*** want to change the PART=ANY operand, 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.

Coding the LOAD Instruction

When you use the LOAD instruction with the Extended Address Mode, you have several options for the PART= operand. The PART= operand indicates the number of the partition in which you want to load the program. The system loads the program in the same partition in which the main program resides if you do not code this operand.

Note: Do not use the PART= operand if the main program loads an overlay program

You can code one of the following for the PART= operand:

- A partition number from 1 to 16.
- PART=ANY to load the program into any available partition. If you specified LOADER=(S,) in the $SRPROF data set, then the loader will try to load the program into one of the static partitions only.
- PART=DYNAMIC to load the program in any available dynamic partition. This option overrides the LOADER=(S,) option in the $SRPROF data set.)
- PART=STATIC to load the program in any available static partition.
- The label of a 1-word data area that contains the partition number. If the data area contains a zero, the system loads the program into any available partition depending on what you specified in ($SRPROF).

In the following example, the system tries to load program PGMA into any available static partition.

Example:

```
LOAD  PGMA,PART=STATIC
```

Note: If you specify the PART= operand as ANY, STATIC, or DYNAMIC, then the order of the partitions into which the system attempts to load a program depends on the LOADER= statement in the IPL configuration data set, $SRPROF.

Notes

Chapter 2. Generate a Tailored Operating System

This chapter provides step-by-step procedures for generating a tailored operating system that supports the EDX Extended Address Mode. The following is a summary of the steps you must perform to generate a tailored operating system with the Extended Address Mode:

- Step 1: Edit $EDXDEF on volume ASMLIB to specify the system definition statements to match your hardware requirements.
- Step 2: Edit $LNKCNTL on volume ASMLIB to include object module SRMGR in your supervisor.
- Step 3: Edit $JOBUTIL procedure data set ($SUPPREP on volume ASMLIB).
- Step 4: Execute $SUPPREP.
- Step 5: Edit $SRPROF on volume ASMLIB to specify the IPL configuration profile.
- Step 6: Using your tailored operating system.
- Step 7: Verify the system generation process (optional).

Generate a Tailored Operating System

Step 1 - Edit $EDXDEF to Match Hardware Configuration

$EDXDEF is a data set on volume ASMLIB containing the system definition statements used in generating the starter system. You must modify $EDXDEF to match your system definition statements. Refer to the *IBM Series/1 Event Driven Executive Installation and System Generation Guide*, SC34-0646, for information on how to edit $EDXDEF to match your hardware configuration.

You also need to edit $EDXDEF to make changes to the SYSTEM and TERMINAL statements for Extended Address Mode support. You can specify up to 16 partitions for the MAXPROG, PARTS, and COMMON operands on the SYSTEM statement. You must specify at least 9 partitions to use Extended Address Mode. You can specify any of the 16 partitions for the INITPRT operand to use for loading $INITIAL.

Changing the SYSTEM Statement

The first four operands of the SYSTEM statement (MAXPROG, PARTS, COMMON, and INITPRT) have the same meaning as described in the *Installation and System Generation Guide* except that for the Extended Address Mode, you can specify up to 16 partitions. For example:

```
SYSTEM MAXPROG=(10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10), X
         PARTS=(32,32,32,32,32,32,32,32,32,32,32,32,32,32,32,32), X
         COMMON=(EDXSVCX,0,1),INITPRT=16
```

The example above defines a 16-partition system with 10 concurrent programs in each partition. The system defines each partition as 64K bytes in size. The system maps the common area into partitions 1 and 3 and tries to load $INITIAL from the IPL volume into partition 16.

Step 1 - Edit $EDXDEF to Match Hardware Configuration *(continued)*

Changing the TERMINAL Statement

With Extended Address Mode, the partition with which a terminal is normally associated can be from 1 to 16. Therefore, the partition number in the PART= operand can be from 1 to 16 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
```

Step 2 - Edit $LNKCNTL to Include Software Support

The $LNKCNTL data set contains all the supervisor object modules needed to generate an operating system. Refer to the *Installation and System Generation Guide* for information on how to edit $LNKCNTL to include the object modules you need.

Including SRMGR

You must include the object module SRMGR if you intend to use Extended Address Mode. 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.

You can include SRMGR in partitions 1 through 8.

Step 2 - Edit $LNKCNTL to Include Software Support *(continued)*

The following is a partial listing of the $LNKCNTL data set showing the module that pertains to Extended Address Mode:

```
*                          .
*                          .
*
*-------------------------------------------------------------------
* EXTENDED ADDRESS MODE    -  MAY BE INCLUDED IN PARTITION 1 TO 8
*-------------------------------------------------------------------
*PART 1
*INCLUDE SRMGR           *29* INCLUDE FOR EXTENDED ADDRESS MODE SUPPORT
*                          .
*                          .
*-------------------------------------------------------------------
*  PROGRAMMING NOTES
*-------------------------------------------------------------------
*                          .
*                          .
*29* INCLUDE FOR EXTENDED ADDRESS MODE SUPPORT.  IF INCLUDED, RLOADER
     WILL AUTOMATICALLY BE INCLUDED; EDXTIMR2 WILL ALSO BE INCLUDED
     AUTOMATICALLY IF EDXTIMER IS NOT INCLUDED.  FOR MORE INFORMATION
     REFER TO THE 'IBM SERIES/1 EVENT DRIVEN EXECUTIVE EXTENDED ADDRESS
     MODE AND PERFORMANCE ANALYZER USER GUIDE', SC34-0591.
```

Figure 1. Partial $LNKCNTL Data Set for Extended Address Mode

Step 3 - Edit $JOBUTIL Procedure File

Refer to the *Installation and System Generation Guide* for information on how to set up the procedure data set, $SUPPREP, to generate your tailored operating system.

Step 4 - Execute $SUPPREP

Refer to the *Installation and System Generation Guide* for information on how to execute $SUPPREP.

Step 5 - Edit $SRPROF - IPL Configuration Profile Data Set

You can edit the $SRPROF configuration profile data set at any time to change the status of a partition, for example, from static to dynamic.

At IPL time, the system reads in $SRPROF and sets up the tables required for managing and performing the dynamic I/O segmentation register allocation and deallocation requests on behalf of the EDX device handlers.

You can edit the $SRPROF data set with the $FSEDIT operator command. Read in $SRPROF from volume ASMLIB, edit it to meet your system's requirements, then write it out to $SRPROF on volume EDX002 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 or you can comment out a line by placing an asterisk in column 1.

If errors occur when the system tries to read this data set or if you fail to edit the data set, the system takes the defaults listed in the example on the following page.

```
*                    EVENT DRIVEN EXECUTIVE                              *
*                  EXTENDED ADDRESS MODE SUPPORT                         *
*                 $SRPROF - IPL CONFIGURATION FILE                       *
*                 VERSION 5 - 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     *
* EXTENDED ADDRESS MODE AND PERFORMANCE ANALYZER USER GUIDE' SC34-0591   *
*                                                                        *
* STATEMENTS WITH AN ASTERISK (*) IN COLUMN 1 ARE IGNORED.               *
*                                                                        *
**************************************************************************
**************************************************************************
* MAY BE 'S', 'D', OR 'L'. 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 ONE     *
* MUST BE STATIC.  ONE AND ONLY ONE OF THE PARTITION'S BANKS OF SEG      *
* REGS MUST BE RESERVED FOR THE LOADER 'L' (THE DEFAULT IS PARTITION 8)  *
**************************************************************************
*
*      12345678
 PARTS=SDDDDDDL
*
**************************************************************************
* 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 1-16) OR IN 'REVERSE' (PARTITIONS 16-1).     *
**************************************************************************
*
 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
*
```

The operands are described as follows:

Operand	*Description*
PARTS=	Can be an S (static), D (dynamic), or L (loader). You can specify only partitions 1 through 8 as static, but partition 1 must be static. You can specify only one partition for the loader. If you do not specify a partition for the loader, the system uses partition 8. Partitions 9–16 can be dynamic only. The default is PARTS=SDDDDDDL.
LOADER=	Specifies two subitems relative to the EDX loader. 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 may 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 partition 16. The default is LOADER=(S,F).
WAITIOSR=	May 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 terminates the task and issues an error message to $SYSLOG. The default is WAITIOSR=Y.

The following example specifies partitions 1–4 as static and partitions 5–7 as dynamic. Partition 8's bank of I/O segmentation registers is reserved for the loader, which makes partition 8 a dynamic partition.

Example: $SRPROF data set.

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

You can set flags in the $TCBFLGS word which will override whatever is coded for WAITIOSR. The following example shows bit settings for $TCBFLGS. An explanation of the numbered items follows the example.

Note: An x indicates that the system ignores the value of the bit. It only checks the 0 and 1 bits indicated below.

Example:

```
          1
XXXX XXX1 XXXX XXXX     CHECK IF I/O IS TO/FROM DYNAMIC/STATIC PARTITION
XXXX XXX0 XXXX XXXX     DON'T CHECK; ISSUE I/O
          2
XXXX XXXX 0XXX XXXX     WAIT FOR ALLOCATION
XXXX XXXX 1XXX XXXX     DON'T WAIT; TASK REMOVED FROM SYSTEM.  ERROR
                        MESSAGE IN $SYSLOG.

DEFAULTS
XXXX XXX1 0XXX XXXX     4-bit mode
```

1 Indicates the $TCBCHK flag. When it is set to 1, the system checks to see if the I/O is to or from a static or dynamic partition. When it is set to 0, the system does not check, it issues the I/O.

2 Indicates the $TCBWAIT flag. When it is set to 0, the system waits until segmentation registers are available to issue I/O. When it is set to 1, the system removes the task and issues an error message to $SYSLOG.

Copy in the TCB equates as follows:

```
COPY TCBEQU
```

The list of equates will include the following four:

```
$TCBCHK  EQU X'0100'
$TCBWAIT EQU X'0080'
          .
          .
$TCBCHKB EQU 7
$TCBWAIB EQU 8
          .
          .
```

Read in $TCBFLGS, turn off the check bit, and put it back into $TCBFLGS as follows:

```
TCBGET   FLAGWORD,$TCBFLGS
SETBIT   FLAGWORD,OFF,+$TCBCHKB
TCBPUT   FLAGWORD,$TCBFLGS
```

Step 6 - Using Your Tailored Operating System

Refer to the *Installation and System Generation Guide* for information on how to use your tailored operating system.

Step 7 - Verify the System Generation Process (Optional)

Refer to the *Installation and System Generation Guide* for information on how to verify that the system generation has been successful.

Notes

Chapter 3. Partition Status and Problem Determination

This chapter contains examples of problem determination tools, return codes, and error messages which pertain to Extended Address Mode support.

Determining the Status of Partitions

You can use the $A operator command to determine whether a partition is static or dynamic. Then use the $CP operator command to change to the partition you need.

For example, if you want to load an application program into a static partition, press the attention key and enter $A to determine if the partition you are in is static. If it isn't, you can press the attention key and enter $A ALL to locate a static partition. Then press the attention key and enter $CP ***n*** where n is the number of the static partition you want.

Refer to the *Operator Commands and Utilities Reference* for information on how to use these operator commands.

Partition Status and Problem Determination

Monitor the Status of System Control Blocks

You can use the $STGUT1 utility to display all the segmentation registers and monitor the status of the system control blocks. Refer to the *Operator Commands and Utilities Reference* for information on how to use this utility.

$DUMP

You can use $DUMP to dump all the CPU segmentation registers and the I/O segmentation registers. Consequently, your stand-alone dump may require four diskettes if you dump to diskette. If you want to avoid having to switch diskettes, you can dump to disk. Refer to the *Operator Commands and Utilities Reference* for information on how to use this utility.

Program Checks

A program check message may appear when you are trying to run a program. The following is an example of a program check:

```
PROGRAM CHECK:
 PLP  TCB  PSW  IAR  AKR  LSR  R0   R1   R2   R3   R4   R5   R6   R7
 3A00 0120 8102 2AD6 0BB0 80D0 0064 3B0A 3B20 3A37 3A34 015C 00B8 0000
```

If you receive an invalid function program check and the IAR (instruction address register) contains the address of CMDSETUP (an entry point in the supervisor module EDXALU), refer to the *IBM Series/1 4956 Processor Model E and Processor Features Description,* GA34-0289, for problem determination information.

Stop Codes

The stop codes for extended address support and their meanings are listed below:

```
CODE   MEANING

BA     LOADBUFR SUPERVISOR MODULE MUST BE INCLUDED IN A MAPPED
       I/O SEG REG AREA (STATIC) IN THE SYSGEN
BA     TPCOM1 SUPERVISOR MODULE MUST BE INCLUDED IN A MAPPED
       I/O SEG REG AREA (STATIC) IN THE SYSGEN
BA     EDXDEFS SUPERVISOR MODULE MUST BE INCLUDED IN A MAPPED
       I/O SEG REG AREA (STATIC) IN THE SYSGEN
BB     DISK ERROR READING $SRPROF DATA SET; RETURN CODE nnn;
       INFORMATION BEFORE ERROR IS TAKEN, THE REST IS DEFAULTED.
BC     LESS THAN 9 PARTITIONS SPECIFIED; 3-BIT MODE OPERATION TAKEN
BD     COMMON AND/OR SUPERVISOR AREA CAN NOT BE STATIC FOR LOADER BANK;
       THE ENTIRE PARTITION # IS DEFINED TO BE DYNAMIC
BE     TIMER SUPPORT NOT INCLUDED IN THE SUPERVISOR
BF     LOADER SUPPORT NOT INCLUDED IN THE SUPERVISOR SYSGEN
C0     THIS IS NOT AN EXTENDED MODE 4-BIT PROCESSOR
C1     INVALID WAITIOSR PARAMETER; DEFAULT TAKEN: Y
C2     INVALID LOADER PARAMETER; DEFAULT TAKEN: (S,F)
C3     INVALID PARTS PARAMETER; DEFAULT TAKEN: SDDDDDDL
```

Error Messages

The following are error messages that the Extended Address Mode function might issue:

$SRPROF PARAMETER IS INVALID: (S,Q)
$SRPROF DATA SET ERROR - INVALID LOADER PARAMETER
DEFAULT TAKEN: (S,F)

Issued by: SRINIT1

Explanation: The LOADER= operand in the $SRPROF data set is invalid.

System Action: The system uses the default.

User Response: See "Step 5 - Edit $SRPROF - IPL Configuration Profile Data Set" on page UG-11 for information on how to code the LOADED operand for the $SRPROF data set.

$SRPROF PARAMETER IS INVALID: SSSSSSSS
$SRPROF DATA SET ERROR - INVALID PARTS PARAMETER
DEFAULT TAKEN: SDDDDDDL

Issued by: SRINIT1

Explanation: The PARTS= operand in the $SRPROF data set is invalid.

System Action: The system uses the default.

User Response: See "Step 5 - Edit $SRPROF - IPL Configuration Profile Data Set" on page UG-11 for information on how to code the PARTS operand for the $SRPROF data set.

$SRPROF PARAMETER IS INVALID: Q
$SRPROF DATA SET ERROR - INVALID WAITIOSR PARAMETER
DEFAULT TAKEN: Y

Issued by: SRINIT1

Explanation: The WAITIOSR operand in the $SRPROF data set is invalid.

System Action: The system uses the default.

User Response: See "Step 5 - Edit $SRPROF - IPL Configuration Profile Data Set" on page UG-11 for information on how to code the WAITIOSR operand for the $SRPROF data set.

COMMON AND/OR SUPERVISOR AREA CANNOT BE STATIC FOR LOADER BANK
REDEFINE THE $SRPROF DATA SET PARTS PARAMETER

Issued by: SRINIT1

Explanation: The loader bank on the PARTS= statement cannot be included in a dynamic partition that has SUPVIO included in the link-control data set.

System Action: The system makes the entire partition DYNAMIC.

User Response: See "Step 5 - Edit $SRPROF - IPL Configuration Profile Data Set" on page UG-11 for information on how to code the $SRPROF data set operands.

DYNAMIC I/O SEG REGS ARE NEEDED FOR 4-BIT MODE
REDEFINE THE $SRPROF DATA SET PARTS= PARAMETER

Issued by: SRINIT2

Explanation: There are no I/O segmentation registers available to the system because there are no dynamic partitions available.

System Action: The system comes up in 3-bit address mode.

User Response: Redefine the PARTS= operand in the $SRPROF data set to include dynamic partitions. See "Step 5 - Edit $SRPROF - IPL Configuration Profile Data Set" on page UG-11 for information on how to code the $SRPROF data set operands.

Error Messages *(continued)*

EDXDEFS SUPERVISOR MODULE MUST BE INCLUDED BEFORE DYNSTART

Issued by: SRINIT1

Explanation: EDXDEFS module was not included in the system generation and is required for extended address support.

System Action: The system comes up in 3-bit address mode.

User Response: Perform another system generation and include EDXDEFS module before the DYNSTART module in the link control data set.

ERROR: THIS IS NOT A 4-BIT ADDRESS MODE PROCESSOR

Issued by: SRINIT1

Explanation: The processor you're trying to IPL with Extended Address Mode support is not a 4-bit processor.

System Action: The system comes up in 3-bit address mode.

User Response: Must use a 4956 model E or 4956-60E processor. To save storage, you should perform another system generation for your current 3-bit processor without SRMGR included.

LESS THAN NINE PARTITIONS DEFINED

Issued by: SRINIT1

Explanation: Fewer than 9 partitions were specified on the PARTS= operand for the SYSTEM statement.

System Action: The system comes up in 3-bit address mode.

User Response: Perform another system generation with 9 to 16 partitions specified on the PARTS= operand of the SYSTEM statement if you want extended address support.

LOADBUFR SUPERVISOR MODULE MUST BE INCLUDED BEFORE DYNSTART

Issued by: SRINIT1

Explanation: LOADBUFR was not included in the system generation and is required for extended address support.

System Action: The system comes up in 3-bit address mode.

User Response: Perform another system generation and include LOADBUFR module before the DYNSTART module in the link-control data set.

PART= MUST BE BETWEEN 1 AND 16

Issued by: $EDXASM

Explanation: The PARTS= operand value for the LOAD instruction is not between 1 and 16.

System Action: The error is flagged and compilation continues.

User Response: Recode the PART= operand using a valid partition number.

RLOADER IS NOT INCLUDED

Issued by: SRINIT1

Explanation: RLOADER was not included and is required for extended address support.

System Action: The system issues stop code BF and comes up in 3-bit address mode. (See "Stop Codes" on page UG-19.)

User Response: Perform another system generation and include RLOADER.

Error Messages *(continued)*

THE SUPERVISOR HAS A COMMON AREA LARGER THAN THE SIZE OF PARTITION 1

Issued by: SRINIT1

Explanation: Partition 1 is smaller than the size of the common area.

System Action: The system comes up in 3-bit address mode.

User Response: Redefine $EDXDEF to reduce the size of the common area or increase the size of partition 1.

TIMER SUPPORT IS NOT INCLUDED

Issued by: SRINIT1

Explanation: EDXTIMR2 was not included and is required for extended address support.

System Action: The system issues stop code BE and comes up in 3-bit address mode. (See "Stop Codes" on page UG-19.)

User Response: Perform another system generation and include EDXTIMR2.

TPCOM1 SUPERVISOR MODULE MUST BE INCLUDED BEFORE DYNSTART

Issued by: SRINIT1

Explanation: TPCOM1 was not included in the system generation and is required for extended address support.

System Action: The system comes up in 3-bit address mode.

User Response: Perform another system generation with TPCOM1 included in the link-control data set.

Chapter 4. Supervisor Module Names (CSECTS)

This chapter contains the names of all the object modules you need to include in your supervisor for Extended Address Mode support. 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. For more information and listings of other EDX CSECTS, refer to the *Installation and System Generation Guide.*

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 set. A module name of $EDXDEF signifies that you must correct an existing definition statement or that you failed to code a required definition statement in the EDXDEFS data set. All other module names signify which supervisor object module must be included in the LINKCNTL data set. Modify the appropriate data set and run the $JOBUTIL procedure again to regenerate your supervisor.

Note: Following each occurrence of $EDXDEF is the name of the definition statement that is either in error or missing.

Supervisor Module Names (CSECTS)

Entry Point	Object Module
$SRTBL	$SRMGR
#ALLSCB	SRMGR
#BMES	$SRMGR
#DEQLBNK	LDRCHK
#DELSCB	SRMGR
#ENQLBNK	LDRCHK
#2K	$SRMGR
BMETBL	$SRMGR
BME0	$SRMGR
BME1	$SRMGR
BME2	$SRMGR
BME3	$SRMGR
BME4	$SRMGR
BME5	$SRMGR
BME6	$SRMGR
BME7	$SRMGR
BSCGSCB	BSCCHK
CURWSCB	$SRMGR
CURWSEG	$SRMGR
DEQLBNK	LDRCHK
DISP	$SRMGR
DSKCHKA	DSKCHK
DSKCHKD	DSKCHK
DYNEND	DYNEND
DYNSTART	DYNSTART
DYNSTRT1	DYNSTRT1
ENQLBNK	LDRCHK
FRSTSCB	$SRMGR
LASTACC	$SRMGR
LOADADS	DSKCHK
LOADBANK	$SRMGR
LOADFLAG	$SRMGR
LOADQCB2	$SRMGR
LOOPCNT	$SRMGR
MAXWSCB	$SRMGR
MAXWSEG	$SRMGR
MAX2K	$SRMGR
MGRADS	$SRMGR
PARTFLAG	$SRMGR
SCB0	SCBTBL
SEGCNT	$SRMGR
SEGCNTX	$SRMGR
SEGMAX	$SRMGR
SEGREG	$SRMGR
SHIFTCNT	$SRMGR
SRINIT1	SRINIT1
SRINIT2	SRINIT2
SRMGRFLG	$SRMGR
SROPEN	SROPEN
START2K	$SRMGR
SUPVIO	SUPVIO
WAITECB	$SRMGR
WORKSCB	SRMGR
WRKSCBA	$SRMGR

Chapter 5. Customizing Your System

Map Supervisor Area for I/O Using SUPVIO

If you include the module SUPVIO in a partition, then the system maps the common area and supervisor area for I/O for both STATIC and DYNAMIC partitions (this overrides the defaults listed below). If you have a supervisor module that performs I/O operations into itself and, therefore, needs to be mapped, you may want to use this feature. If you include the supervisor and SUPVIO modules in a static partition, the I/O operation will work the same way as it did for Version 4.0. If you include them in a dynamic partition, however, you must reset the bits on the task issuing the I/O for it to work the same way as it did for Version 4.0 (see "Step 5 - Edit $SRPROF - IPL Configuration Profile Data Set" on page UG-11).

For a DYNAMIC partition, the system maps the common area, supervisor area, and user area as dynamic. For a STATIC partition, the system maps the common area and the user area as static and the supervisor area as dynamic. The one exception is for partition 1 where the system maps the supervisor (up to the end of the system definition statements) as static.

The following table shows the mapping if you include SUPVIO and the defaults when you do not include SUPVIO. An explanation of the numbered items follows the example.

Map Supervisor Area for I/O Using SUPVIO *(continued)*

MAPPING DEFAULTS WITHOUT SUPVIO:

Part #	Part Type	Common Area	Supervisor Area	User Area
[1] 1	STATIC	STATIC	[2] STATIC/DYNAMIC	STATIC
2 - 8	STATIC	STATIC	DYNAMIC	STATIC
[3] 2 - 8	DYNAMIC	DYNAMIC	DYNAMIC	DYNAMIC
9 - 16	DYNAMIC	DYNAMIC	DYNAMIC	DYNAMIC

MAPPING WITH SUPVIO INCLUDED:

Part #	Part Type	Common Area	Supervisor Area	User Area
1 - 8	STATIC	STATIC	STATIC	STATIC
[4] 2 - 8	DYNAMIC	STATIC	STATIC	DYNAMIC

[1] Partition 1 must be STATIC and you cannot use it for the LOADER.

[2] The supervisor area is STATIC up to the end of the system definition statements in partition 1.

[3] You can reserve only one partition for the LOADER, and the partition must be between 2 and 8.

[4] If you include SUPVIO in a partition, you cannot reserve that same partition for the LOADER. If you try to do this, the system issues a stop code and sends an error message to $SYSLOG, and the entire partition is dynamic.

The following figures illustrate what happens to a dynamic or static partition in which you have included SUPVIO. The shaded portions illustrate areas that are ***not*** mapped for I/O segmentation registers.

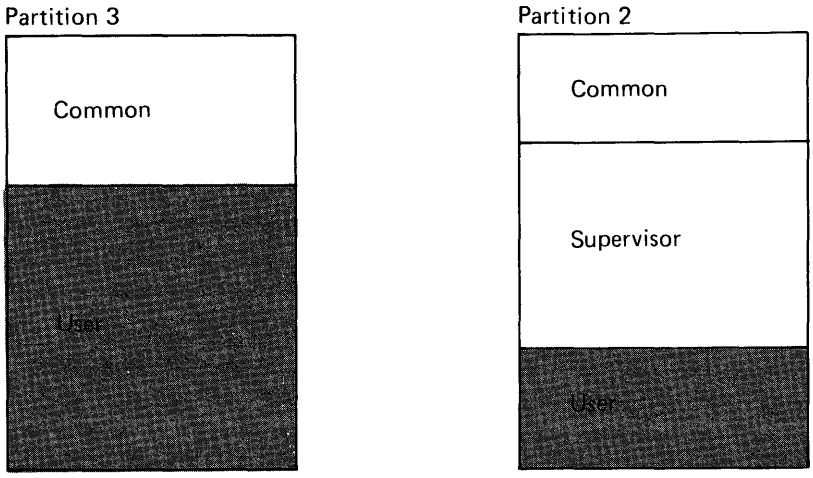


Figure 2. SUPVIO in Dynamic Partitions.

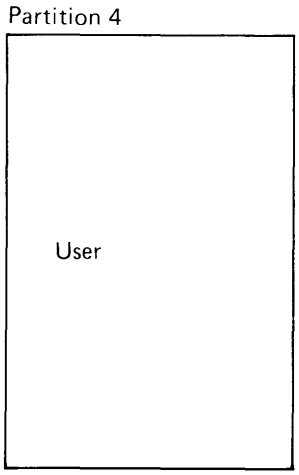


Figure 3. SUPVIO in Static Partitions.

The following is a partial listing of the $LNKCNTL data set showing the modules that pertain to SUPVIO:

```
                          .
                          .
                          .
*-------------------------------------------------------------------
* SUPERVISOR SUPPORT     - MUST BE FIRST AND IN PARTITION 1
*-------------------------------------------------------------------
*PART 1
                          .
                          .
                          .
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*
*-------------------------------------------------------------------
                          .
                          .
                          .
*-------------------------------------------------------------------
* SUPERVISOR CODE BEING MOVED OUT OF
* PARTITION 1 MUST BE MOVED TO HERE
*-------------------------------------------------------------------
*PART 2
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 3
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 4
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 5
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 6
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 7
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*PART 8
*INCLUDE SUPVIO          *28*   MAKE SUPERVISOR AND COMMON AREA STATIC
*
*-------------------------------------------------------------------
*  PROGRAMMING NOTES
*-------------------------------------------------------------------
                          .
                          .
                          .
*28* MAKES ALL THE COMMON AND SUPERVISOR AREA OF EACH PARTITION THAT
*    SUPVIO IS INCLUDED IN STATIC (ONLY VALID FOR EXTENDED ADDRESS
*    MODE SUPPORT).
```

Figure 4. Partial $LNKCNTL Data Set Showing SUPVIO

Map Supervisor Area for I/O Using DYNSTART and DYNEND

DYNSTART and DYNEND are two additional modules you can include in the $LNKCNTL data set (see "Edit $LINKCNTL to Include DYNSTART or DYNEND" on page UG-33 for an example). You can use these modules, either together or separately, to make part of the supervisor in a static partition mapped for I/O segmentation registers. You can include DYNSTART in partitions 2 – 8 and DYNEND in partitions 1 – 8. However, if you include DYNEND in conjunction with DYNSTART in any partition, then you ***must*** include DYNEND in every partition.

Notes:

1. If you limit the size of the unmapped I/O segmentation register area within your static partitions, you limit the number of I/O segmentation registers that the system can use for the partitions you defined as static in the $SRPROF data set.

2. If you include SUPVIO in a partition, it overrides DYNSTART or DYNEND.

Figure 5 illustrates two static partitions (which you defined as such in the $SRPROF data set). The figure on the left does not include DYNSTART; the shaded region shows that the entire supervisor area is unmapped. The figure on the right shows the same partition with DYNSTART included; the shaded region shows that only the supervisor area following DYNSTART remains unmapped.

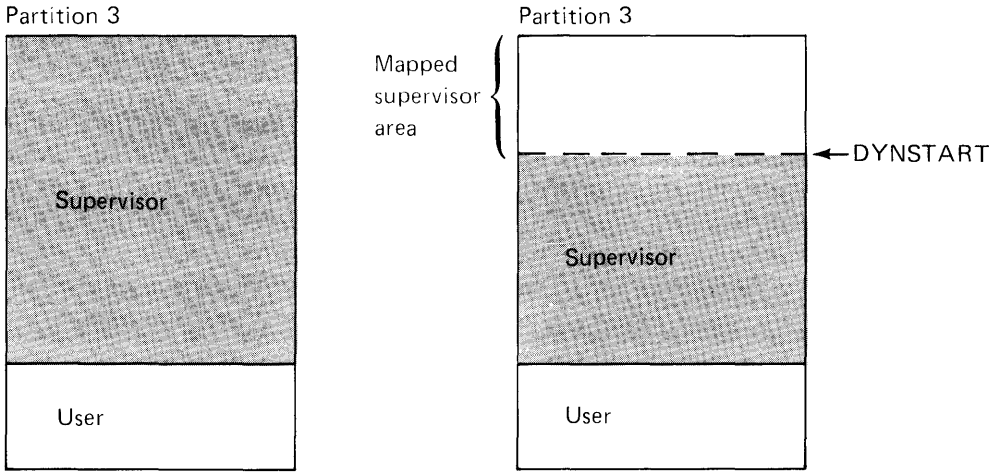


Figure 5. Unmapped Supervisor Areas for Static Partitions.

Figure 6 illustrates two additional static partitions. The one on the left does not include DYNSTART; the shaded region shows that the entire supervisor area is unmapped. The figure on the right shows the same partition with DYNSTART included; the shaded region shows that only the supervisor area following DYNSTART remains unmapped.

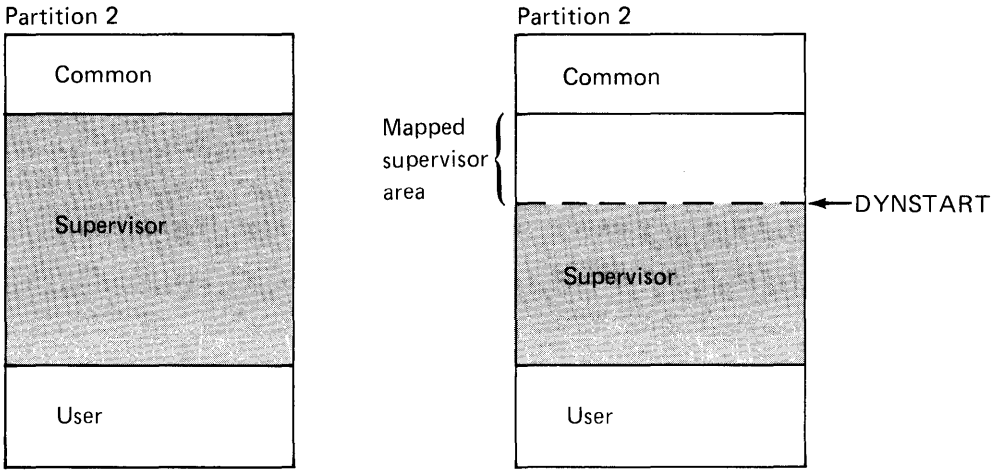


Figure 6. DYNSTART in Static Partitions.

Figure 7 illustrates two static partitions. The figure on the left does not include DYNSTART or DYNEND; the shaded region shows that the entire supervisor area is unmapped. The figure on the right shows the same partition with DYNEND included by the user; the shaded region shows that only the supervisor area preceding DYNEND remains unmapped. Figure 8 on page UG-33 shows the same partition with both DYNSTART and DYNEND included by the user; the shaded region shows that now only the supervisor area between DYNSTART and DYNEND remains unmapped.

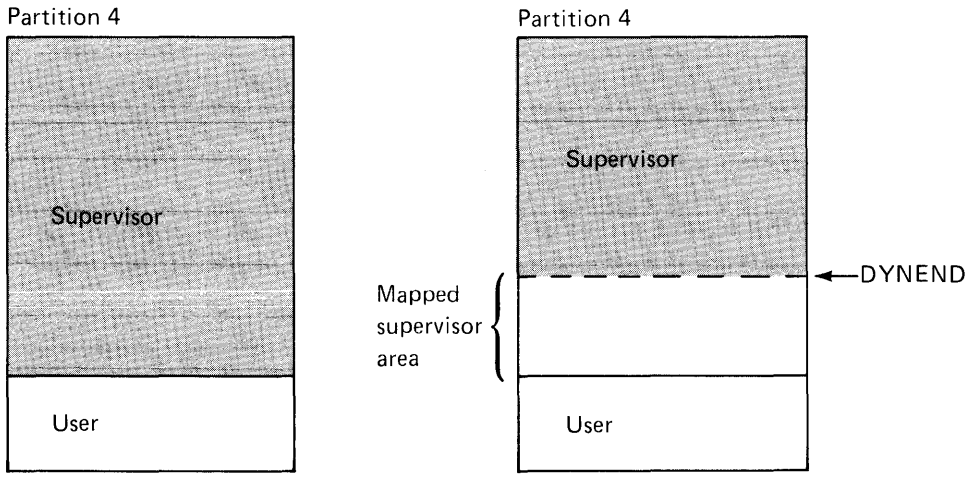


Figure 7. DYNEND in a Static Partition.

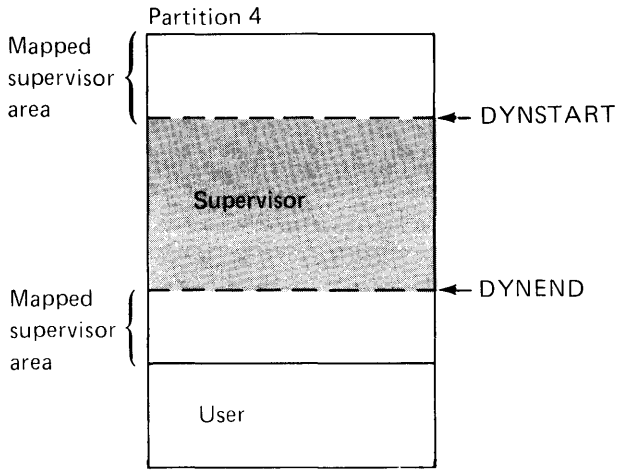


Figure 8. DYNSTART and DYNEND in a Static Partition.

Edit $LINKCNTL to Include DYNSTART or DYNEND

The following three examples show partition 2 in the $LINKCNTL data set. Example 1 illustrates partition 2 without DYNSTART included. Example 2 illustrates partition 2 with DYNSTART included. Example 3 illustrates partition 2 with DYNSTART and DYNEND included. Partition 2 is defined as static in the PARTS= operand of $SRPROF.

Example 1: Partition 2 without DYNSTART included.

```
PART 2
 INCLUDE DISKIO          *3*     BASIC DISKETTE SUPPORT
 INCLUDE D49624          *3*     4962/4964 DISK(ETTE) SUPPORT
 INCLUDE D4963A          *3*     4963/4967/DDSK-30 DISK SUPPORT
 INCLUDE D4966A          *3*     4965/4966 DISKETTE SUPPORT
*INCLUDE D1024           *3,21*  1024 BYTES/SECTOR DISKETTE SUPPORT
*INCLUDE D4969A          *3*     BASIC TAPE SUPPORT
*INCLUDE D4969A          *3*     BASIC TAPE SUPPORT
```

Example 2: Partition 2 with DYNSTART included.

```
 PART 2
 INCLUDE DISKIO          *3*    BASIC DISKETTE SUPPORT
 INCLUDE D49624          *3*    4962/4964 DISK(ETTE) SUPPORT
 INCLUDE MYBUFF1   ===========  THIS WILL BE I/O SEG REG MAPPED AREA
 INCLUDE DYNSTART  ===========  START OF I/O SEG REG UNMAPPED AREA
 INCLUDE D4963A          *3*    4963/4967/DDSK-30 DISK SUPPORT
 INCLUDE D4966A          *3*    4965/4966 DISKETTE SUPPORT
*INCLUDE D1024           *3,21* 1024 BYTES/SECTOR DISKETTE SUPPORT
*INCLUDE D4969A          *3*    BASIC TAPE SUPPORT
*INCLUDE D4969A          *3*    BASIC TAPE SUPPORT
```

Example 3: Partition 2 with DYNSTART and DYNEND included.

```
 PART 2
 INCLUDE DISKIO          *3*    BASIC DISKETTE SUPPORT
 INCLUDE D49624          *3*    4962/4964 DISK(ETTE) SUPPORT
 INCLUDE MYBUFF1   ===========  THIS MODULE WILL BE MAPPED
 INCLUDE DYNSTART  ===========  START OF I/O SEG REG UNMAPPED AREA
 INCLUDE D4963A          *3*    (UNMAPPED) 4963/4967/DDSK-30 DISK SUPPORT
 INCLUDE D4966A          *3*    (UNMAPPED) 4965/4966 DISKETTE SUPPORT
 INCLUDE DYNEND    ===========  END OF I/O SEG REG UNMAPPED AREA
 INCLUDE MYBUFF2   ===========  THIS MODULE WILL BE MAPPED
*INCLUDE D1024           *3,21* 1024 BYTES/SECTOR DISKETTE SUPPORT
*INCLUDE D4969A          *3*    BASIC TAPE SUPPORT
*INCLUDE D4969A          *3*    BASIC TAPE SUPPORT
```

Notes:

1. MYBUFF1 and MYBUFF2 are illustrations of statically-defined user I/O buffer areas.

2. Since you are using DYNEND in conjunction with DYNSTART in partition 2, you have to include DYNEND in all other partitions as well.

Part 2. EDX Performance Analyzer

Part 2 contains information about the EDX Performance Analyzer. The Performance Analyzer is divided into two functions: The System Analyzer, covered in Chapter 6, and the Program Analyzer, covered in Chapter 7. You can use these two functions to identify the major performance problem areas on your system and to monitor any modifications you make to try to improve that performance.

Chapter 8 contains information on ways to improve your system's overall performance once you have analyzed the problem areas.

Chapter 9 contains a list of the error messages for both the System Analyzer and the Program Analyzer.

Note: The numbers that appear in the generated reports are not necessarily exact.

Notes

Chapter 6. Analyzing System Performance

Using the System Analyzer

You can use the System Analyzer to monitor how your system uses I/O resources for any time period. $S1PSYS is the main system monitor. You can use it to track all task dispatches, I/O interrupts, and wait states and record them in a data set.

$S1PSYSR is the report generator for the System Analyzer. It sorts the data gathered by $S1PSYS and generates the various reports that you specify (see "Generating a Report" on page UG-40).

$S1PSYS Requirements

$S1PSYS has three requirements:

- You must allocate a data set to be used as a statistics file for storing the information gathered by the monitor program.
- You must have EDX timer support.
- You must have disk support.

The size of your data set depends on the amount of activity on your system and the length of time you specify for the data to be collected while the analyzer is running. The information that the analyzer gathers is saved in 50-byte logical records within its internal buffers. The system creates a 50-byte record each time a task ends and another 50-byte record for each data set the task accesses. At checkpoint time the analyzer groups these 50-byte records together by fives into a 256-byte physical record, then writes the physical records to the data set.

Loading the System Analyzer ($S1PSYS)

You can load $S1PSYS into any partition as follows:

```
> $L $S1PSYS
 DSNAME(NAME,VOLUME):  DS1,EDX002
LOADING $S1PSYS  57P,00:00:29, LP= 7E00, PART=1
```

You can include the statistics data set name and the volume on the same line when you load $S1PSYS. If you don't, the system prompts you for the data set name and volume. If you do not specify a volume, the system automatically uses the IPL volume.

If $S1PSYS is using the data set for the first time, the system displays the following message. Respond Y if you want $S1PSYS to use the data set or N if you want to end $S1PSYS:

```
DS HAS NOT PREVIOUSLY BEEN USED
AS AN S1PSYS DATA SET.
IS IT OK TO USE IT NOW (Y/N)?  Y
--- INITIALIZING STATISTICS FILE; PLEASE STAND BY. ---
```

Once you have loaded $S1PSYS successfully, you receive the following message:

```
  EDX SYSTEM MONITOR
--- MONITOR ACTIVE ---
```

You control the analyzer by using the following attention commands:

Command	*Description*
> #ACI	Specifies the interval (in minutes) at which to take checkpoints (when the analyzer writes all current data to the statistics data set). The valid range is 1–30 inclusive. The default is 15 minutes. If you do not enter a value on the same line as the #ACI command, the system prompts you to set an interval. If you enter an interval, the system then displays the interval you set. If you do not enter an interval, the system displays the current interval.

```
> #ACI
AUTO-CHECKPOINT INTERVAL (MINUTES):  10
AUTO-CHECKPOINT INTERVAL SET TO -    10 MINUTES
```

> #AOV Specifies whether or not your statistics data set will overflow automatically when the monitor program encounters the end of the data set.

If you set auto-overflow to ON, $S1PSYS does not warn you when an overflow occurs but starts back at the beginning of your data set and continues monitoring. The data already existing previous to the overflow is lost as the analyzer writes new data.

```
> #AOV
AUTO-OVERFLOW ON/OFF?  ON
--- AUTO-OVERFLOW SET ON ----
```

If you set auto-overflow to OFF, the default, $S1PSYS does warn you when an overflow condition exists:

```
*** DISK DATA SET OVERFLOW ***
WOULD YOU LIKE TO CONTINUE MONITORING (Y/N)?
```

If you respond Y, $S1PSYS will start back at the beginning of your data set and continue monitoring. Once again, the data already existing previous to the overflow will be lost. If you want to obtain a copy of that data before it is lost, load $S1PSYSR (the report generator) on another terminal and print out the data set contents before responding Y to the overflow prompt.

If you respond N to the prompt, $S1PSYS asks if you want to generate a report (see #END explanation).

> #CKP Performs a checkpoint immediately and displays the following message:

```
--- CHECKPOINT COMPLETE ---
```

> #END Ends $S1PSYS ($C operator command will not work). After you enter the command, the system prompts you as follows:

```
> #END
WOULD YOU LIKE TO GENERATE A REPORT (Y/N)?
```

If you respond N, the monitor operation ends. If you respond Y, as in the example, the system ends $S1PSYS and loads the $S1PSYS report generator:

```
$S1PSYS ENDED AT 00:03:36

$S1PSYS REPORT GENERATOR

COMMAND (?):
```

Generating a Report

$S1PSYSR, the report generator, formats and prints the data that $S1PSYS records.

Note: The numbers in the generated reports are not necessarily exact.

You can load $S1PSYSR with the #END command or with the $L operator command as shown below:

```
> $L $S1PSYSR
 DSNAME(NAME,VOLUME):  DS1,EDX002
LOADING $S1PSYSR      20P,00:41:54, LP=0000, PART=2

$S1PSYS REPORT GENERATOR

COMMAND (?):
```

The example above uses the default value for the dynamic storage parameter. This parameter determines the amount of storage that $S1PSYSR uses for sorting the data from the $S1PSYS statistics data set (DS1 in this case). The amount of dynamic storage you need depends on the amount of activity on the system when the analyzer stored its information and the length of time the analyzer ran.

The default for dynamic storage is 2048 bytes (256 bytes = 1 page). If you want to allocate less or more dynamic storage, you must specify it when you load $S1PSYSR. The format is:

```
> $L $S1PSYSR,volume,dynamic-storage dsname
```

For example, to specify 1024 bytes of dynamic storage, load $S1PSYSR as follows:

```
> $L $S1PSYSR,EDX002,1024 DS1
```

If your dynamic storage area is too small, $S1PSYSR issues the following message:

```
*** INSUFFICIENT DYNAMIC STORAGE FOR SORT ***
    DO YOU WISH TO USE A WORK DATA SET (Y/N)?
```

If you respond N, the operation is cancelled. If you respond Y, the system prompts you for DSNAME (NAME,VOLUME), then resumes sorting using the work data set that you specify.

$S1PSYSR Commands

To display the $S1PSYSR commands at your terminal, enter a question mark in response to the prompting message COMMAND (?).

```
COMMAND (?):  ?

             DATA SELECTION COMMANDS:
             ------------------------
LL     --  SPECIFY LOWER DATE/TIME LIMIT
UL     --  SPECIFY UPPER DATE/TIME LIMIT
LCP    --  SPECIFY LOWER CHECKPOINT LIMIT
UCP    --  SPECIFY UPPER CHECKPOINT LIMIT
RUN    --  INCLUDE RECORDS FROM LAST S1PSYS RUN (DEFAULT)
ALL    --  INCLUDE ALL RECORDS IN STATFILE

             REPORT SELECTION COMMANDS:
             --------------------------
PRD    --  SELECT PROGRAM DETAIL REPORT (DEFAULT)
PRDG   --  PROGRAM DETAIL REPORT -- GENERIC PROGRAM NAME
PRDNG  --  PROGRAM DETAIL REPORT -- NONGENERIC PROGRAM NAME
CPT    --  SELECT CHECKPOINT SUMMARY REPORT ON USER TERMINAL
CPP    --  SELECT CHECKPOINT SUMMARY REPORT ON PRINTER
PRS    --  SELECT PROGRAM SUMMARY REPORT
DSS    --  SELECT DATA SET SUMMARY REPORT

             PROGRAM CONTROL:
             ----------------
LIST   --  SPECIFY OUTPUT DEVICE NAME
EN     --  TERMINATE PROGRAM EXECUTION

COMMAND (?):
```

After $S1PSYSR displays the commands, it prompts you again for the command of your choice. The Data Selection Commands allow you to specify how much of the gathered data will be used in the analyzer reports. For example, you might load $S1PSYS and let it run two hours. During the two-hour period, the analyzer monitors your system and saves the information in the statistics data set. After you stop the monitoring and end the analyzer, load $S1PSYSR (the report generator). If you decide you are only interested in the data gathered in the last hour that the monitor ran, use the LL command and set the lower limit for the reporting period to one hour after the monitor started. Until you enter another data selection command, $S1PSYSR will produce reports that use only the data from the last hour.

Each command and its explanation is presented on the following pages in the order it appears on your screen.

LL — Specify Lower Date/Time Limit

Use the LL command to specify a lower limit for the date and time of the reporting period.

Note: If you use LL in conjunction with LCP, the one you specify ***last*** will be the one that $S1PSYS uses in selecting data.

Example: Set the lower limit for the reporting period.

```
COMMAND (?):  LL
DATE(M/D/Y OR M/D):  07/23/84
TIME(H.M):  10:00
LOWER DATE/TIME LIMIT IS 07/23/84 10:00:00
```

If you do not set the date but you do set the time, the system uses "today's" date and the time you set. If you do not set the time, the system uses 00:00. If you do not set the date and the time, the system uses 00/00/00 for the date and 00:00 for the time.

UL — Specify Upper Date/Time Limit

Use the UL command to specify an upper limit for the date and time of the reporting period. If you do not specify a limit, the system uses the time and date when you ended $S1PSYS.

Note: If you use UL in conjunction with UCP, the one you specify ***last*** will be the one that $S1PSYS uses in selecting data.

Example: Set the upper limit for the reporting period.

```
COMMAND (?):  UL
DATE(M/D/Y OR M/D):  07/24/84
TIME(H.M):  10:00
UPPER DATE/TIME LIMIT IS 07/24/84 10:00:00
```

If you do not set the date but you do set the time, the system uses "today's" date and the time you set. If you do not set the time, the system uses 23:59. If you do not set the year, the system uses the current year.

LCP — Specify Lower Checkpoint Limit

Use the LCP command to specify the lower limit for the checkpoint number. For instance, if you did not want the period of time between checkpoints 1 – 5 listed on the final report, you could set the lower limit to 6. If you do not specify a limit, the system includes the period of time starting with the first checkpoint. Zero (0) is not a valid limit.

Note: If you use LL in conjunction with LCP, the one you specify ***last*** will be the one that $S1PSYS uses in selecting data.

Example: Set the lower checkpoint interval.

```
COMMAND (?): LCP
CHECKPOINT NUMBER: 6

LOWER CHECKPOINT LIMIT SET TO 6

COMMAND (?):
```

You can list the checkpoints by using the CPT or CPP commands.

UCP — Specify Upper Checkpoint Limit

Use the UCP command to specify the upper limit for the checkpoint number. For instance, if the system made 100 checkpoints but you do not want the period of time ***after*** checkpoint 56 listed on the final report, specify 56 as the upper limit. If you do not specify a limit, the system includes the period of time up to the last checkpoint. Zero (0) is not a valid limit.

Note: If you use UL in conjunction with UCP, the one you specify ***last*** will be the one that $S1PSYS uses in selecting data.

Example: Set the upper checkpoint interval.

```
COMMAND (?): UCP
CHECKPOINT NUMBER: 56

UPPER CHECKPOINT LIMIT SET TO 56

COMMAND (?):
```

RUN — Include Records from Last Run

Use the RUN command to specify that the report include data from the last execution of $S1PSYS only. This is the default.

Example: Include data from the last execution of $S1PSYS only.

```
COMMAND (?):  RUN
```

ALL — Include All Records in the Statistics File

Use the ALL command to specify that the report include data from all the records contained in the statistics data set. For example, you can run the system analyzer one hour a day for a week and save all the data in the same statistics data set. However, unless you specify the ALL command when you generate a report, you will only get the data from the last run of $S1PSYS. If you specify ALL, the report will contain the entire week's data.

Example: Include data from all records in the data set.

```
COMMAND (?):  ALL
```

PRD — Select Program Utilization Detail Report

Use the PRD command to specify that you want a program utilization detail report for ***all*** programs. This report is a combination of two other types of reports: the program summary and the data set summary (see the PRS and DSS command explanations). If you want to list only specific programs, use the generic (PRDG) or nongeneric (PRDNG) commands.

Example: Select and print the program utilization detail report.

```
COMMAND (?):  PRD

--- SORTING FOR REPORT; PLEASE STAND BY. ---

--- REPORT PRINTING ---
--- REPORT COMPLETE ---
```

The system prints the report on your system printer unless you specify otherwise with the LIST command.

The following is an example of the program utilization detail report. Some of the field headings in this example are abbreviated. An explanation of the report fields follows the example.

```
DSNAME:  DS1,EDX002

         EDX PERFORMANCE ANALYZER - SYSTEM REPORT

                  EDX PROGRAM UTILIZATION DETAIL     10/05/84 12:35:36  to  10/05/84 12:48:56

                                                                                                  AVG.
                                                                                                  ACC.
     START                PROGRAM        TCB              ELAPS.          INT.   DS   VOLUME              TIME
  DATE     TIME     P  NAME      #PGS  ADDR  CPU TIME     TIME    %CPU  %CPU  NAME NAME    DA  CNT   (MS)

10/05/84 12:35:36  1  PXRAM       8   OFDE  0:04:865      0:16   12.6   0.0  DS1  VOL001  03  1535  23.50
                                                                            DSA  VOL002  03  1683  24.31
                                                                            DSB  VOL003  03  1923  30.44
10/05/84 12:37:43  7  PXMENU     79   09B8  0:14:038     12:23    1.8   0.0  DS2  VOL116  03  1515  31.15
10/05/84 12:39:07  5  $DISKUT2   89   13B0  0:13:043      1:51   11.8   0.0  DS3  VOL001  02  1602  84.70
10/05/84 12:41:45  8  $DISKUT1   86   CC6C  0:10:622       :42   25.    0.0  DS4  MYVOL   03  1738  26.66
                                    .
                                    .
                                    .
                         <CHECKPOINT OVHD.>          0:00.028                0.0
                         <SYSTEM WAIT TIME>         12:40.014               99.9
                         <INTERRUPT SERVICE>         0:00.385                0.0
                                                    ----------              ------
                                                    13:20.154               99.9
```

Report Field ***Description***

Start Date/Time
The starting date and time of the program. If the program started before the reporting period started, this field shows the start of the reporting period.

P The number of the partition into which the program was loaded.

Program Name
The name of the program that was executing. The name <SYSTEM> indicates a system program with a TCB but no name, such as the EDX loader.

#Pgs The size of the program in pages (1 page = 256 bytes). The <OV> indicates an EDX overlay program. The <ST> indicates an EDX subtask associated with the program.

TCB Addr The task control block address for the data on this line. Multiple copies of the same program or multiple tasks within a program produce separate TCB addresses and separate report lines.

CPU Time The total CPU time used by this execution of the task.

Elaps. Time The elapsed time (hh:mm:ss), within the reporting period, for this task. A "+" following the time indicates that the task did not end during the reporting period.

% CPU The program's CPU utilization expressed as a percentage of the program's elapsed time. If the time is too small, the field will be blank.

Int. % CPU The program's CPU utilization expressed as a percentage of the machine's potential CPU time used by this task during the reporting interval.

Ds Name The name of each data set accessed by the task. A data set name $$ indicates EDX loader accesses to the volume directory.

Volume Name The name of the volume where the data set resides.

Da The device address for each data set.

Cnt The total count of accesses to each data set.

Avg. Acc. Time
The average access time for all accesses to the data set, expressed in milliseconds (ms).

<CHECKPOINT OVHD.>

CPU Time The CPU time required by the monitor to take checkpoints.

Int. % CPU Percentage of the total CPU time represented by this overhead.

<SYSTEM WAIT TIME>

CPU Time The total wait time during the reporting interval. Although this total is displayed under CPU time, it is wait time, not CPU time.

Int. % CPU Percentage of total interval the system was in a wait state.

<INTERRUPT SERVICE>

CPU Time CPU time assigned to the system overhead to handle interrupts that cannot be assigned to a particular task.

Int. % CPU The percentage of CPU time used for interrupt services.

PRDG — Print Program Detail Report (Generic Name)

Use the PRDG command to print only programs beginning with the generic name that you specify.

Example: Print only the programs beginning with a dollar sign ($).

```
COMMAND (?):  PRDG

GENERIC NAME:  $

--- SORTING FOR REPORT; PLEASE STAND BY. ---

--- REPORT PRINTING ---
--- REPORT COMPLETE ---
```

The report prints on your system printer unless you specify otherwise with the LIST command.

PRDNG — Print Program Detail Report (Nongeneric Name)

Use the PRDNG command to print programs that do ***not*** begin with the generic name that you specify.

Example: Print only the programs that do not begin with a dollar sign ($).

```
COMMAND (?):  PRDNG

NONGENERIC NAME:  $

--- SORTING FOR REPORT; PLEASE STAND BY. ---

--- REPORT PRINTING ---
--- REPORT COMPLETE ---
```

The report prints on your system printer unless you specify otherwise with the LIST command.

CPT — Display Checkpoint Summary Report on Terminal

Use the CPT command to display the checkpoints on your terminal screen.

Example: Display checkpoint information on the terminal.

```
COMMAND (?): CPT

--- SCANNING FOR CHECKPOINT RECORDS; PLEASE STAND BY. ---

CP #    DATE AND TIME
----    -------------
  1     07/23/84  00:02:58
  2     07/23/84  00:03:28

COMMAND (?):
```

CPP — Print Checkpoint Summary Report on Printer

Use the CPP command to print the checkpoints on your printer.

Example: Print checkpoint information on the printer.

```
COMMAND (?): CPP

--- SCANNING FOR CHECKPOINT RECORDS; PLEASE STAND BY. ---

CP #    DATE AND TIME
----    -------------
  1     07/23/84  00:02:58
  2     07/23/84  00:03:28

COMMAND (?):
```

PRS — Print Program Summary Report

Use the PRS command to print the program summary report. The report is a summary of all the programs that executed during a specified period of time. It lists and summarizes each program on a separate line.

Example: Print the program summary report.

```
COMMAND (?): PRS

--- SORTING FOR REPORT; PLEASE STAND BY. ---

--- REPORT PRINTING ---
--- REPORT COMPLETE ---

COMMAND (?):
```

The report prints on your system printer unless you specify otherwise with the LIST command.

The following is an example of the program summary report. An explanation of the report fields follows the example.

```
DSNAME:  DS1,EDX002

         EDX PERFORMANCE ANALYZER - SYSTEM REPORT

                 EDX PROGRAM SUMMARY
        05/13/84  12:35:38  TO  05/14/84  12:35:56

PROGRAM            # OF                    ELAPSED            I/O
NAME       #PGS    RUNS     CPU TIME        TIME     %CPU     COUNT

$DISKUT2    45       10      5:34:241        2:54     1.3      4342
STRESS      33        4      0:44:836        6.29    10.5     11345
                                  .
                                  .
                                  .
MYPROG      57        6      4:29:133        1:23     3.9        59
```

Heading	*Description*
Program Name	The name of the program.
#Pgs	The size of the program in pages (1 page = 256 bytes).
# of Runs	The number of times this program ran during the recording period.
CPU Time	The total CPU time used by all executions of the program.
Elapsed Time	The total of the elapsed times for all executions of the program within the reporting period.
% CPU	The percentage of the elapsed time that the program was using the CPU. The total will not always add up to 100% because of rounding.
I/O Count	The number of disk I/O operations performed by all the executions of the program.

DSS — Print Data Set Summary Report

Use the DSS command to print the data set summary report. The data set summary report is a summary by data set and volume of all accesses performed to each data set.

Example: Print the data set summary report.

```
COMMAND (?): DSS

--- SORTING FOR REPORT; PLEASE STAND BY. ---

--- REPORT PRINTING ---
--- REPORT COMPLETE ---

COMMAND (?):
```

The report prints on your system printer unless you specify otherwise with the LIST command. The following is an example of the data set summary report. An explanation of the report fields follows the example.

```
DSNAME:  DS1,EDX002

         EDX PERFORMANCE ANALYZER - SYSTEM REPORT

                EDX DATA SET SUMMARY
       05/13/84  23:16:55  TO  05/14/84  23:25:58

     VOLUME     DATA SET                  % OF ACCESS     AVG. ACC.
DA   NAME       NAME             COUNT   VOLUME   DEVICE   TIME (MS)

44   DISK04     DATA01             433     21.0     20.2       67.92
44   DISK04     DATA02             432     21.0     20.2       68.45
44   DISK04     DATA03             701     34.0     32.7       41.55
44   DISK04     DATA04             495     24.0     23.7       47.16
                               -------             -----      ------
TOTAL FOR VOLUME DISK04          2061             96.2       54.08

44   EDX002     DS1                  4    100.0      0.2       35.00
                               -------             -----      ------
TOTAL FOR VOLUME EDX002              4              0.2       35.00

44              DISK04              12     15.6      0.6       47.66
44              $$                  65     84.4      3.0       21.70
                               -------             -----      ------
TOTAL FOR VOLUME                    77              3.6       25.75

TOTAL FOR DEVICE 44               2142                        53.02
```

Report Field	*Description*
DA	Device address.
Volume Name	The name of the volume where the data set resides.
Data Set Name	The name of the data set. The $$ that appears in this field usually refers to accesses made to the volume directories. When the volume field is blank and the data set name field contains the volume name, that volume's directory was accessed.
Count	The total number of accesses to the data set.
% of Access Volume	The number of accesses to the data set expressed as a percentage of all accesses to the volume.
% of Access Device	The number of accesses to the data set expressed as a percentage of all accesses to the disk device.
Avg. Acc. Time	The average access time for all accesses to the data set, expressed in milliseconds (ms).

LIST — Specify Output Device

Use the LIST command to specify the output device where you want your data printed.

Example: Specify MYPRNT as the output device.

```
COMMAND (?):  LIST
OUTPUT DEVICE NAME:  MYPRNT

COMMAND (?):
```

EN — End the Report Generator Program

Use the EN command to end the report generator program.

```
COMMAND (?):  EN

$S1PSYSR ENDED AT 00:19:47
```

Notes

Chapter 7. Analyzing Program Performance

Using the Program Analyzer

You can use the Program Analyzer to monitor and analyze the use of resources within a program. $S1PPRG is the main program monitor. You must allocate a data set (using the $DISKUT1 utility) that $S1PPRG can use to store the information it gathers. You can include that data set name and the volume on the same line when you load $S1PPRG. If you don't, the system prompts you for the data set name and volume. If you do not specify a volume, the system automatically uses the IPL volume.

Loading the Program Analyzer ($S1PPRG)

You can load $S1PPRG into any partition as follows:

```
> $L $S1PPRG
 DSNAME  (NAME,VOLUME):  DF1,EDX002
LOADING $S1PPRG       33P,00:31:13, LP=7E00, PART=1

EDX PROGRAM ANALYZER
```

The example above uses the default value for the dynamic storage parameter. This parameter determines the amount of storage that $S1PPRG uses for program monitoring. The default for dynamic storage is 256 bytes (256 bytes = 1 page). If you want to allocate less or more dynamic storage, you must specify it when you load $S1PPRG.

The format for specifying dynamic storage is:

```
> $L $S1PPRG,volume,dynamic-storage dsname
```

For example, to specify 512 bytes of dynamic storage, load $S1PPRG as follows:

```
> $L $S1PPRG,EDX002,512 DF1
```

After the system loads $S1PPRG, it prompts you for the name of the program you want to monitor, the partition where you want the program loaded, and the terminal where you want the program displayed. In the following example, the system loads your program into partition 4 and uses $SYSLOG as the display terminal. You can continue to use the terminal where you loaded $S1PPRG.

```
PROGRAM NAME:  MYPROG
PARTITION (DEFAULT IS CURRENT PARTITION):  4
TERMINAL (DEFAULT IS CURRENT TERMINAL):  $SYSLOG
MYPROG      75P     WILL BE LOADED WHEN SCANNING STARTS.

COMMAND (?):
```

If the target program is already loaded in partition 4, the system displays the following message (after the PARTITION prompt) and lists the load point(s):

```
ALREADY LOADED AT 4300 5400
DO YOU WANT TO LOAD ANOTHER COPY (Y/N)?
```

If you respond Y to the prompt, the system prompts for the terminal. If you respond N and the program has only one load point, the system will use the copy that is already loaded. If you respond N and there is more than one load point listed (as in the example -- 4300 and 5400), the system prompts you for which load point you want:

```
PROGRAM LOAD POINT:  5400
```

$S1PPRG Commands

To display the commands at your terminal, enter a question mark in response to the prompting message COMMAND (?).

```
COMMAND (?):  ?

GO        INITIATE PROGRAM SCANNING
INT       SPECIFY SCAN INTERVAL (DEFAULT = 100 MS.)
AT        SPECIFY SCAN RANGE (DEFAULT = ENTIRE PROGRAM)
RNG       DISPLAY CURRENTLY SELECTED SCAN RANGE
INC       INCLUDE PROGRAM WAIT TIME IN REPORT GRAPH
SEP       SEPARATE WAIT TIME FROM REPORT GRAPH (DEFAULT)
EN        TERMINATE PROGRAM EXECUTION

COMMAND (?):
```

After the system displays the commands, it prompts you again for the command of your choice. Each command and its explanation is presented on the following pages in alphabetical order.

AT — Specify Scan Range

Use the AT command to specify the range of addresses that the system monitors within the target program. Each address is 1 to 4 hexadecimal digits, indicating an address relative to the start of the target program. Odd numbers round down to even ones. The default range is the entire program.

Since dynamic storage is divided into double-word count bins (256 bytes of dynamic storage = 64 count bins), each count bin is associated with one of the address ranges in the target program. Each time $S1PPRG scans the target program, it determines the address range where the target program is executing and increments the corresponding count bin. (The count bin values are used for the report generator.)

In the following example, the specified address range is from address 0100 to 0500. $S1PPRG scans the entire target program, but the area from 0100 to 0500 is much more detailed.

Example: Set the scanning address range.

```
COMMAND (?):  AT
FROM ADDR:  0100
  TO ADDR:  0500
RANGE 1:     FROM 0000    TO 00FE    BY 0100
RANGE 2:     FROM 0100    TO 0500    BY 0010
RANGE 3:     FROM 0502    TO 4AFE    BY 45FE

COMMAND (?):
```

EN — End Program Execution

Use the EN command to end the report generator sampling program.

Example: End sampling program.

```
COMMAND (?): EN

DO YOU WANT TO GENERATE A REPORT (Y/N)?
```

If you specify N to the "DO YOU WANT TO GENERATE A REPORT" prompt, the program ends. If you specify Y, $S1PPRG loads the report generator and then ends.

```
$S1PPRG  ENDED AT 00:22:57

$S1PPRG REPORT GENERATOR

COMMAND (?):
```

See "Generating a Report" on page UG-58 for a description of how to use the report generator.

GO — Initiate Program Scanning

Use the GO command to start the monitoring operation for the program you specified when you loaded $S1PPRG. This command deletes any previous sampling results from the data set.

Example: Load MYPROG and start sampling.

```
COMMAND (?): GO
LOADING MYPROG    75P,00:39:05, LP= 1000, PART= 4
SCANNING STARTED AT 00:39:05.  ENTER ATTN (>) #END TO END SCANNING.
```

If the terminal you specified is busy, the system prompts you as follows:

```
TERMINAL BUSY.  WOULD YOU LIKE TO WAIT (Y/N)?
```

If you respond Y, the system waits for that terminal and starts scanning only when the terminal is no longer busy. If you respond N, the system uses the terminal where you loaded $S1PPRG.

Note: Once you specify GO, you can use only the attention commands to control the monitoring operation (see "Controlling $S1PPRG Execution" on page UG-58).

INC — Include Program Wait Time in the Report

Use the INC command to include CPU time and non-CPU wait times in the counting and on the report graph (see example under LP command).

Example: Include CPU time and non-CPU wait times on report.

```
COMMAND (?):  INC

COMMAND (?):
```

INT — Specify Scan Interval

Use the INT command to specify the scanning interval in milliseconds. Valid interval values range from 1 to 2000 inclusive. The default is 100 milliseconds.

Example: Set the scanning interval.

```
COMMAND (?):  INT
SAMPLE INTERVAL:  7
SCANNING INTERVAL SET TO -    7 MS.

COMMAND (?):
```

RNG — Display Currently Selected Scan Range

Use the RNG command to display the target address range and the size of the locations for which the analyzer is currently taking sample counts.

Example: Display the current scan range.

```
COMMAND (?):  RNG
RANGE 1:      FROM 0000     TO 00FE     BY 0100
RANGE 2:      FROM 0100     TO 0500     BY 0010
RANGE 3:      FROM 0502     TO 4AFE     BY 45FE

COMMAND (?):
```

SEP — Separate Wait Time from the Report

Use the SEP command to include only the CPU time in the counting and on the report. This is the default.

Example: Include only CPU time on the report.

```
COMMAND (?):  SEP

COMMAND (?):
```

Controlling $S1PPRG Execution

You control execution of $S1PPRG with the following attention commands:

> #STOP Stops sampling. The system displays the message SCANNING STOPPED AT hh:mm and prompts you for another command.

> #END Ends the program. The system gives you the option of generating a report (see EN command description above).

Note: The data that the analyzer gathers remains in the dynamic storage area until either you enter the attention #END command or the program you are monitoring ends. Then the analyzer writes the data to the disk data set.

Generating a Report

$S1PPRGR, the report generator, formats and prints the data that the analyzer records.

Note: The numbers in the generated reports are not necessarily exact.

You can load $S1PPRGR in three ways: With the EN or attention #END commands of $S1PPRG, or with the $L operator command. The following example shows how to load it with the $L operator command:

```
> $L $S1PPRGR,EDX002,512 DF1
LOADING $S1PPRGR     20P,00:41:54, LP= 1000, PART= 2

$S1PPRG REPORT GENERATOR

COMMAND (?):
```

The parameters are the same as for the analyzer. The dynamic storage value (512 in the above example) must be at least equal to the value you specified when you loaded the analyzer. The default is 256 bytes.

$S1PPRGR Commands

To display the report generator commands at your terminal, enter a question mark in response to the prompting message COMMAND (?).

```
COMMAND (?):  ?

OUTPUT DEVICE CONTROL
---------------------
LP       GENERATE REPORT ON PRINTER
LT       GENERATE REPORT ON TERMINAL

REPORT FORMAT CONTROL
---------------------
HS       SPECIFY HORIZONTAL SCALING VALUE (DEFAULT = 100%)
ZS       SET ZERO SUPPRESSION ON OR OFF (DEFAULT = ON)

PROGRAM CONTROL
---------------------
LIST     SPECIFY OUTPUT DEVICE NAME
EN       TERMINATE PROGRAM EXECUTION

COMMAND (?):
```

After the system displays the commands, it prompts you again for the command of your choice. Each command and its explanation is presented on the following pages in the order it appears on your screen.

LP — Generate Report on the Printer

Use the LP command to print the report on either the default output device ($SYSPRTR) or an output device you specified with the LIST command.

Example: Generate the report on the system printer.

```
COMMAND (?):  LP
--- REPORT PRINTING ---
--- REPORT COMPLETE ---

COMMAND (?):
```

Generating a Report *(continued)*

The following example of the report shows the default horizontal scale of 100% on the graph. It also shows the default "zero suppression ON" (ZS command). With ZS on, instead of listing all the addresses from 0000 to 036E separately, the system groups them together (indicated by the > sign) to save considerable space.

```
PROGRAM NAME:  MYPROG (2 TASKS)
SCANNED FROM 05/18/84 10:15:05 UNTIL 05/18/84 10:18:41
SCANNING INTERVAL WAS 100 MS.   1712 SCANS WERE DONE.
DSNAME:  DF1, EDX003

15.7% TIME SPENT WAITING FOR CPU
70.1% TIME SPENT IN NON-CPU WAITS IN PRIMARY TASK
75.7% TIME SPENT IN NON-CPU WAITS IN TASK #2
38.5% TIME SPENT IN EXECUTION (INCLUDED IN GRAPH)

                          ------------------------------------------100%
0000 > 036E    0.0%   |
0370 - 0376    3.6%   | *****
0378 - 037E    3.7%   | *****
0380 - 0386    2.8%   | **
0388 - 038E    3.9%   | *****
0390 - 0396    1.2%   | *
0398 > 04CE    0.0%   |
04D0 - 04D6    0.4%   | *
04D8 > 05D6    0.0%   | *
05D8 - 05DE    0.2%   | *
05E0 > 179E    0.0%   | *
17A0 - 17A6    4.1%   | *******
17A8 - 1836    0.0%   |
1838 - 183E    2.7%   | **
1840 - 1846    0.0%   |
1848 - 184E    3.6%   | *****
1850 - 1856    5.1%   | *********
1858 - 185E    4.2%   | *******
1860 - 1866    3.0%   | ***
1868 - 1BFE    0.0%   |
              ------   ------------------------------------------100%
              38.5%
```

Interpreting the Report

With this report you can see the results of monitoring MYPROG, a program with two tasks. The scanning started at 10:15:05 on 05/18/84. MYPROG executed for 3 minutes 36 seconds (subtract the beginning and ending scan times). 15.7% of that time was delays caused by higher-priority tasks using the processor when MYPROG was trying to run.

All percentages are based on the total elapsed time, that is, on the total number of scans. The primary task spent 70.1% of the total execution time in wait instructions (I/O delays, terminal waits, ENQs, and so on), while Task #2 spent 75.7% of the total execution time in wait instructions. Both tasks were simultaneously in wait instructions most of the time. Between the two tasks, a total of 38.5% of the elapsed time was spent executing instructions.

The report graph breaks down the execution time by instruction location. If you look at the graph in the example, you can see that 14.0% of the total elapsed time was spent executing instructions between 0370 and 038E (3.6 + 3.7 + 2.8 + 3.9 = 14.0). Since no single address location accounted for more than 5.1% of the total CPU time, you might find it useful to specify a smaller horizontal scale and regenerate the report (see the HS command explanation for an example).

The example graph does not show wait time. If you specified the INC command, the graph would include the wait times (shown in the upper part of the report) broken done by location.

LT — Generate Report on the Terminal

Use the LT command to list the report at your terminal. The output will look exactly the same as it does when you specify the LP command.

HS — Specify Horizontal Scaling Value

Use the HS command to control the horizontal scale of the report graph for greater clarity. You can specify a value from 0% – 100% (the default).

Example: Specify a horizontal scale of 6%.

```
COMMAND (?):  HS
% VALUE:  6

COMMAND (?):
```

If you specify the LP command, the output appears as follows:

```
                    ------------------------------------------- 6%
0000 > 036E   0.0%  |
0370 - 0376   3.6%  | *************************
0378 - 037E   3.7%  | **************************
0380 - 0386   2.8%  | ********************
0388 - 038E   3.9%  | ***********************************
0390 - 0396   1.2%  | **********
0398 > 039E   0.0%  |
04D0 - 04D6   0.4%  | ****
04D8 > 05D6   0.0%  | *
05D8 - 05DE   0.2%  | *
05E0 > 179E   0.0%  | *
17A0 - 17A6   4.1%  | ************************************
17A8 - 1836   0.0%  |
1838 - 183E   2.7%  | ***********************
1840 - 1846   0.0%  |
1848 - 184E   3.6%  | *****************************
1850 - 1856   5.1%  | *********************************************
1858 - 185E   4.2%  | ************************************
1860 - 1866   3.0%  | ************************
1868 - 1BFE   0.0%  |
              ------  ------------------------------------------- 6%
              38.5%
```

Note: If you specify 0% for the horizontal scale, the report lists a number representing a percentage of the actual number of scans for that address range.

ZS — Set Zero Suppression On or Off

Use the ZS command to control how the graph groups adjacent instruction locations that do not contain sample counts (0.0% on the graph). Specify ON (the default) to group the addresses together. Specify OFF to separate the addresses on the graph. (See the explanation of the LP command for an example.)

Example: Specify zero suppression off.

```
COMMAND (?):  ZS
ON/OFF:  OFF

COMMAND (?):
```

LIST — Specify Output Device Name

Use the LIST command to control where the LP command will list your report output. You must specify the device name.

Example: Specify MYPRNTR as the output device.

```
COMMAND (?): LIST
OUTPUT DEVICE NAME: MYPRNTR

COMMAND (?):
```

EN — End Program Execution

Use the EN command to end the report generator program.

Example: End the report generator.

```
COMMAND (?): EN

$S1PPRGR ENDED AT 00:29:43
```

Notes

Chapter 8. Improving System Performance

Once you have used the system and program analyzers to identify performance problems, you can use the information you gathered to improve your system's performance. Improving performance may be as simple as finding and eliminating the one major bottleneck on your system. However, you may find that you need a detailed analysis, extensive reprogramming, or even a change to the architecture of your system. A thorough understanding of the application you are monitoring is essential.

Set Up Controls

When you do performance tuning, you must establish a "control" group. Then you can determine if your efforts are actually improving the performance of your system.

For example, if you have a transaction-based system, you can set up a control group of ten transactions of a particular type. Using the Performance Analyzer, you would then monitor the group for data set access speeds, response times, and number of disk I/O operations. Then each time you change something on your system, monitor the same group to see the improvement those changes made.

Improving System Performance

Improvement Techniques

Chapter 8 of the *Customization Guide* contains several techniques for improving your system's performance. The chapter describes how to:

- Get faster access to data sets and volumes
- Define your DISK statements
- Specify fixed-head volumes
- Improve disk and tape I/O performance
- Speed up the $COMPRES and $COPYUT1 utilities
- Decrease $EDXASM compilation time
- Reduce program load time

Analyzing System Reports

You can use the various reports generated by the System Analyzer and change your system accordingly.

1. Analyze your "Data Set Summary Report" (see the DSS command description in "$S1PSYSR Commands" on page UG-41 for an example of the report) to determine the volume which has the most disk activity. Put that volume in the center of the disk. Put the next most heavily used volume on one side and the third most heavily used volume on the other side, and so on.

   For instance, you normally allocate volume EDX002 first after initializing a disk. That places EDX002 at the far end of the disk, but in most cases you use this volume more heavily than any other. To improve access time, place this volume in the center of the disk as follows.

   a. Before allocating EDX002, allocate a volume that is one half the size of your disk.

   b. Allocate EDX002. (You might also consider making this volume large enough to hold the required system modules only.)

   c. Delete the volume you allocated initially.

You can also use the Data Set Summary Report to analyze data set activity. Then you should rearrange the data sets on each volume so that the most heavily used data sets are side by side. If the average time required to access data sets on one volume is significantly higher than on another volume, you may have initialized the disk with "write verify" on. Write verify doubles the time required for each write operation.

If your system contains more than one disk drive, place the most heavily used volume in the center of one disk drive, the second most heavily used volume in the center of another disk drive, and so on. You can also place your program-type data sets on one drive and your data-type data sets on another.

2. Instead of putting all your application programs and data sets on your IPL volume, try keeping only EDX functions on that volume. Create a separate volume for your application programs, another for application job streams, another for menus, and as many as you need for data.

3. You can make all your volumes "performance" volumes to achieve the best processing speeds. The Data Set Summary Report contains the number of attempts the system made to open volumes other than performance volumes. Under the totals for each volume is a reference to a volume name $$DDyy (yy is the device address) and a data set $$. If the system accesses only performance volumes, $$ does not appear on the report. If it does appear, you know that the system is accessing nonperformance volumes. The number of times $$ is refer to is an indication of your performance degradation.

You can use the Data Set Summary Report to determine the frequency of program loads. Every time you load an application program, the system reads $LOADER into storage. If you place $LOADER and executable programs onto the volume MEMDSK (created with the $MEMDISK utility), you can reduce your load times as described below.

Reducing Program Load Time

The $MEMDISK utility enables you to use unmapped storage as a disk. It creates a volume in storage named MEMDSK. By placing data or programs on this volume, you can reduce access time. Keep in mind, however, that since MEMDSK is part of the memory system, you will lose the volume in the event of a power failure or when you IPL.

By placing the EDX loader ($LOADER) on this "disk," $LOADER also becomes storage-resident, which decreases program load time. Normally, you would have to run the $MEMDISK and $COPYUT1 utilities interactively to perform this function. However, through the use of a $INITIAL program and virtual terminals, you can perform this same function as part of the IPL process. The following virtual terminal sample program shows how this is done.

The following program first loads $MEMDISK and allocates a disk volume (MEMDSK) in unmapped storage. Then the program loads $COPYUT1 and copies the loader to the storage disk just created. Finally, the program loads $MEMDISK again to indicate that the loader now resides on MEMDSK.

Example:

```
MDISK       PROGRAM     START
START       EQU         *
            ENQT        B
*****************************************************************
*                                                               *
*   LOAD $MEMDISK AND ISSUE THE COMMANDS TO ALLOCATE THE        *
*   STORAGE-RESIDENT DISK.  ANSWER1 CONTAINS THE COMMANDS.      *
*                                                               *
*****************************************************************
            LOAD        $MEMDISK,PARM,LOGMSG=NO,EVENT=ECB
            TCBGET      RC,$TCBCO
            IF          (RC,NE,-1),GOTO,ERR     LOAD OK?
            ENQT        A                       SYNC SIDE
PASS1       MOVEA       #1,ANSWER1+2
            DO          4,TIMES
               DO       UNTIL,(RC,EQ,8)
                READTEXT  LINE,MODE=LINE
                TCBGET    RC,$TCBCO
               ENDDO
             PRINTEXT   (0,#1)                  SEND REPLY
             ADD        #1,6                    POINT TO NEXT
            ENDDO
            READTEXT    LINE,MODE=LINE          PGM ENDED MSG
            WAIT        ECB
            ENQT        B
*****************************************************************
*                                                               *
*   LOAD $COPYUT1 AND ISSUE THE COMMANDS TO COPY $LOADER        *
*   FROM DISK TO STORAGE.  ANSWER2 CONTAINS THE COMMANDS.       *
*                                                               *
*****************************************************************
            LOAD        $COPYUT1,LOGMSG=NO,EVENT=ECB
            TCBGET      RC,$TCBCO
            IF          (RC,NE,-1),GOTO,ERR     LOAD OK?
            ENQT        A                       SYNC SIDE
PASS2       MOVEA       #1,ANSWER2+2
            DO          8,TIMES
               DO       UNTIL,(RC,EQ,8)
                READTEXT  LINE,MODE=LINE
                TCBGET    RC,$TCBCO
               ENDDO
             PRINTEXT   (0,#1)                  SEND REPLY
             ADD        #1,10                   POINT TO NEXT
            ENDDO
            READTEXT    LINE,MODE=LINE          PGM ENDED MSG
            WAIT        ECB
            ENQT        B
```

```
*****************************************************************
*                                                               *
*   LOAD $MEMDISK AND ISSUE THE COMMANDS TO SET THE LOADER      *
*   AS STORAGE RESIDENT.  ANSWER3 CONTAINS THE COMMANDS.        *
*                                                               *
*****************************************************************
           LOAD        $MEMDISK,PARM,LOGMSG=NO,EVENT=ECB
           TCBGET      RC,$TCBCO
           IF          (RC,NE,-1),GOTO,ERR      LOAD OK?
           ENQT        A                        SYNC SIDE
PASS3      MOVEA       #1,ANSWER3+2
           DO          2,TIMES
              DO       UNTIL,(RC,EQ,8)
               READTEXT  LINE,MODE=LINE
               TCBGET    RC,$TCBCO
              ENDDO
            PRINTEXT   (0,#1)                   SEND REPLY
            ADD        #1,4                     POINT TO NEXT
           ENDDO
           READTEXT    LINE,MODE=LINE           PGM ENDED MSG
           WAIT        ECB
           DEQT
           GOTO        DONE
ERR        EQU         *
           DEQT
           PRINTEXT    'ERROR ON LOAD, RC= '
           PRINTNUM    RC                       RETURN CODE
           PRINTEXT    SKIP=1
DONE       PROGSTOP
PARM       DATA        F'0'
RC         DATA        F'0'
ECB        ECB
A          IOCB        CDRVTA                   SYNC=YES
B          IOCB        CDRVTB
ANSWER1    EQU         *
           TEXT        'IV',LENGTH=4            ALLOCATE MEMDSK
           TEXT        '500',LENGTH=4           SIZE
           TEXT        '100',LENGTH=4           NBR DATA SETS
           TEXT        'EN',LENGTH=4
ANSWER2    EQU         *
           TEXT        'Y',LENGTH=8             EDX002 SOURCE?
           TEXT        'N',LENGTH=8             EDX002 TARGET?
           TEXT        'MEMDSK',LENGTH=8        TARGET VOLUME
           TEXT        'Y',LENGTH=8             SOURCE/TARGET OK?
           TEXT        'CM',LENGTH=8            COPY MEMBER
           TEXT        '$LOADER',LENGTH=8       SOURCE MEMBER
           TEXT        '*',LENGTH=8             TARGET MEMBER
           TEXT        'EN',LENGTH=8
ANSWER3    EQU         *
           TEXT        'SL',LENGTH=2            SET LOADER
           TEXT        'EN',LENGTH=2
LINE       TEXT        LENGTH=80                I/O CHANNEL
           ENDPROG
           END
```

After you compile this program, you must name the program $INITIAL if you want it loaded at IPL. (Refer to *Customization Guide* for the procedure.)

Note: To use $MEMDISK, you must include the module STORMGR in your supervisor during system generation (refer to the *Installation and System Generation Guide*). (For more information on the $MEMDISK utility, refer to the *Operator Commands and Utilities Reference*; for information and examples on the use of virtual terminals, refer to the *Event Driven Executive Language Programming Guide* and the *Language Reference.*)

The $MEMDISK utility temporarily requires 28 pages (7K) of storage in the partition where you load the utility. In addition, as long as the MEMDSK volume is defined to the system, $MEMDISK support acquires and holds the following storage resources:

- Two pages in partition one (1 page = 256 bytes).
- One page in the partition containing your disk I/O support modules.
- A 2K block of storage (on a 2K boundary) in the highest partition from which the system can obtain it or from the partition you specify as the fifth parameter in the example below.

You can load $MEMDISK with the $L operator command and use it interactively as with many other EDX utilities. You can also load $MEMDISK by using the EDL LOAD instruction with a command passed as a parameter, but you will not be able to copy $LOADER as in the previous example. However, if you do not have virtual terminals or if you want to be able to specify the partition for the required 2K block of storage, you may want to use the LOAD instruction.

The following example shows the "allocate" function: the IV command of $MEMDISK. The IV command is the only command for which all five parameters apply. All other commands need only the first parameter. An explanation of the five parameters follows the example:

Example:

```
            .
            .
            LOAD        $MEMDISK,PARM,LOGMSG=NO,EVENT=ECB
            .
            .
PARM        DC          C'IV 00 800 0100 06'
            .
            .
```

Parameter	*Description*
IV	The command to be issued (in this case, ALLOCATE). Valid commands are IV, DV, SL, RL, SD, RD.
00	The system attempts to allocate MEMDSK with the number of records you specify in the third parameter (in this example, 800). If this is not possible, $MEMDISK does not allocate any records and the system issues a return code of 100. If you code ***01*** for this second parameter, the system attempts to allocate MEMDSK with the number of records you specify in the third parameter. If this is not possible, $MEMDISK allocates the volume MEMDSK as large as possible. If you code ***01*** for this second parameter and ***0*** for the third parameter, $MEMDISK allocates the volume MEMDSK as large as possible.
800	The number of records requested for the MEMDSK volume.
0100	The number of data sets you want the MEMDSK volume to accommodate.
06	The partition from which the system will take the 2K block of storage. If you don't specify this parameter, the system takes storage from the highest partition where it is available.

Analyzing Individual Programs

The Program Analyzer provides you with information that you can use in individual program tuning. By reviewing the Program Analyzer reports, you can see where a certain program spends most if its time in execution. This information is used differently for each program. For instance, if an EDL program performs several I/O operations to disk, the Program Analyzer shows that the program spent more time performing these operations than for other operations (which is normal). However, if a particular READ or WRITE operation takes longer to execute than other READ/WRITE operations, you might have a resource conflict between this program and another program running on the system. Both programs may be trying to gain exclusive use of the disk, and one has to wait for the other to release the disk.

You can use these reports differently for each application. Therefore, it is important that you have thorough knowledge of what the applications are designed to do before you use the Program Analyzer.

Notes

Chapter 9. Performance Analyzer Error Messages

The following are error messages that the System Analyzer and Program Analyzer may produce. They are listed in alphabetical order only.

ADDRESS RANGE SPECIFICATION ERROR

Issued by: $S1PPRGR

Explanation: Data set contains invalid data.

System Action: Ends the analysis operation.

User Response: Reload $S1PPRG and $S1PPRGR.

EOF ENCOUNTERED BEFORE THE END OF DATA

Issued by: $S1PPRGR

Explanation: Data set contains invalid data.

System Action: Ends the analysis operation.

User Response: Reload $S1PPRG and $S1PPRGR.

INSUFFICIENT DYNAMIC STORAGE ALLOCATED

Issued by: $S1PPRGR

Explanation: You failed to allocate enough dynamic storage on the $S1PPRGR load.

System Action: Ends the report program.

User Response: Reload $S1PPRGR with storage amount at least equal to that specified when you loaded $S1PPRG.

INSUFFICIENT SPACE ON WORKFILE

Issued by: $S1PSYSR

Explanation: The specified work data set was too small.

System Action: Cancels current operation. Returns to COMMAND (?).

User Response: Use a larger work data set or restart the program with more dynamic storage.

INVALID OPTION -- XX

Issued by: $S1PPRGR

Explanation: You specified a nonexistent command.

System Action: Reprompts you to select another command.

User Response: Reenter a valid command or a question mark (?) for a list of the valid commands.

NO DYNAMIC STORAGE ALLOCATED

Issued by: $S1PPRGR

Explanation: You failed to allocate dynamic storage when you loaded $S1PPRGR.

System Action: Ends the report program.

User Response: Reload $S1PPRGR with storage amount at least equal to that specified when you loaded $S1PPRG.

NO RECORDS ON FILE IN THE SPECIFIED TIME INTERVAL

Issued by: $S1PSYSR

Explanation: The statistics data set contains no records.

System Action: No report will be produced. Returns to COMMAND (?).

User Response: Rerun the monitor program and then the report program; correctly specify the reporting time interval.

<OVERFLOW>

Issued by: $S1PSYSR

Explanation: Indicates that the monitoring program ran out of buffer space while collecting data.

System Action: Ends monitoring operation. Returns to COMMAND (?).

User Response: Set a lower checkpoint interval next time you run the System Analyzer ($S1PSYS).

TIMER SUPPORT NOT PRESENT ON THIS SYSTEM

Issued by: $S1PSYS

Explanation: Either your system does not have timer support or you failed to include timer support.

System Action: $S1PSYS ends.

User Response: Include timer support for your system.

Glossary of Terms and Abbreviations

This glossary defines terms and abbreviations used in the Series/1 Event Driven Executive software publications. All software and hardware terms pertain to EDX. This glossary also serves as a supplement to the *IBM Data Processing Glossary*, GC20-1699.

$SYSLOGA, $SYSLOGB. The name of the alternate system logging device. This device is optional but, if defined, should be a terminal with keyboard capability, not just a printer.

$SYSLOG. The name of the system logging device or operator station; must be defined for every system. It should be a terminal with keyboard capability, not just a printer.

$SYSPRTR. The name of the system printer.

abend. Abnormal end-of-task. Termination of a task prior to its completion because of an error condition that cannot be resolved by recovery facilities while the task is executing.

ACCA. See asynchronous communications control adapter.

address key. Identifies a set of Series/1 segmentation registers and represents an address space. It is one less than the partition number.

address space. The logical storage identified by an address key. An address space is the storage for a partition.

application program manager. The component of the Multiple Terminal Manager that provides the program management facilities required to process user requests. It controls the contents of a program area and the execution of programs within the area.

application program stub. A collection of subroutines that are appended to a program by the linkage editor to provide the link from the application program to the Multiple Terminal Manager facilities.

asynchronous communications control adapter. An ASCII terminal attached via #1610, #2091 with #2092, or #2095 with #2096 adapters.

attention key. The key on the display terminal keyboard that, if pressed, tells the operating system that you are entering a command.

attention list. A series of pairs of 1 to 8 byte EBCDIC strings and addresses pointing to EDL instructions. When the attention key is pressed on the terminal, the operator can enter one of the strings to cause the associated EDL instructions to be executed.

backup. A copy of data to be used in the event the original data is lost or damaged.

base record slots. Space in an indexed file that is reserved for based records to be placed.

base records. Records are placed into an indexed file while in load mode or inserted in process mode with a new high key.

basic exchange format. A standard format for exchanging data on diskettes between systems or devices.

binary synchronous device data block (BSCDDB). A control block that provides the information to control one Series/1 Binary Synchronous Adapter. It determines the line characteristics and provides dedicated storage for that line.

Glossary of Terms and Abbreviations

block. (1) See data block or index block. (2) In the Indexed Method, the unit of space used by the access method to contain indexes and data.

block mode. The transmission mode in which the 3101 Display Station transmits a data data stream, which has been edited and stored, when the SEND key is pressed.

BSCAM. See binary synchronous communications access method.

binary synchronous communications access method. A form of binary synchronous I/O control used by the Series/1 to perform data communications between local or remote stations.

BSCDDB. See binary synchronous device data block.

buffer. An area of storage that is temporarily reserved for use in performing an input/output operation, into which data is read or from which data is written. See input buffer and output buffer.

bypass label processing. Access of a tape without any label processing support.

CCB. See terminal control block.

central buffer. The buffer used by the Indexed Access Method for all transfers of information between main storage and indexed files.

character image. An alphabetic, numeric, or special character defined for an IBM 4978 Display Station. Each character image is defined by a dot matrix that is coded into eight bytes.

character image table. An area containing the 256 character images that can be defined for an IBM 4978 Display Station. Each character image is coded into eight bytes, the entire table of codes requiring 2048 bytes of storage.

character mode. The transmission mode in which the 3101 Display Station immediately sends a character when a keyboard key is pressed.

cluster. In an indexed file, a group of data blocks that is pointed to from the same primary-level index block, and includes the primary-level index block. The data records and blocks contained in a cluster are logically contiguous, but are not necessarily physically contiguous.

COD (change of direction). A character used with ACCA terminal to indicate a reverse in the direction of data movement.

cold start. Starting the spool facility by erasing any spooled jobs remaining in the spool data set from any previous spool session.

command. A character string from a source external to the system that represents a request for action by the system.

common area. A user-defined data area that is mapped into the partitions specified on the SYSTEM definition statement. It can be used to contain control blocks or data that will be accessed by more than one program.

completion code. An indicator that reflects the status of the execution of a program. The completion code is displayed or printed on the program's output device.

constant. A value or address that remains unchanged thoughout program execution.

controller. A device that has the capability of configuring the GPIB bus by designating which devices are active, which devices are listeners, and which device is the talker. In Series/1 GPIB implementation, the Series/1 is always the controller.

conversion. See update.

control station. In BSCAM communications, the station that supervises a multipoint connection, and performs polling and selection of its tributary stations. The status of control station is assigned to a BSC line during system generation.

cross-partition service. A function that accesses data in two partitions.

cross-partition supervisor. A supervisor in which one or more supervisor modules reside outside of partition 1 (address space 0).

data block. In an indexed file, an area that contains control information and data records. These blocks are a multiple of 256 bytes.

data record. In an indexed file, the records containing customer data.

data set. A group of records within a volume pointed to by a directory member entry in the directory for the volume.

data set control block (DSCB). A control block that provides the information required to access a data set, volume or directory using READ and WRITE.

data set shut down. An indexed data set that has been marked (in main storage only) as unusable due to an error.

DCE. See directory control entry.

device data block (DDB). A control block that describes a disk or diskette volume.

direct access. (1) The access method used to READ or WRITE records on a disk or diskette device by specifying their location relative the beginning of the data set or volume. (2) In the Indexed Access Method, locating any record via its key without respect to the previous operation. (3) A condition in terminal I/O where a READTEXT or a PRINTEXT is directed to a buffer which was previously enqueued upon by an IOCB.

directory. (1) A series of contiguous records in a volume that describe the contents in terms of allocated data sets and free space. (2) A series of contiguous records on a device that describe the contents in terms of allocated volumes and free space. (3) For the Indexed Access Method Version 2, a data set that defines the relationship between primary and secondary indexed files (secondary index support).

directory control entry (DCE). The first 32 bytes of the first record of a directory in which a description of the directory is stored.

directory member entry (DME). A 32-byte directory entry describing an allocated data set or volume.

display station. An IBM 4978, 4979, or 3101 display terminal or similar terminal with a keyboard and a video display.

DME. See directory member entry.

DSCB. See data set control block.

dynamic storage. An increment of storage that is appended to a program when it is loaded.

end-of-data indicator. A code that signals that the last record of a data set has been read or written. End-of-data is determined by an end-of-data pointer in the DME or by the physical end of the data set.

ECB. See event control block.

EDL. See Event Driven Language.

emulator. The portion of the Event Driven Executive supervisor that interprets EDL instructions and performs the function specified by each EDL statement.

end-of-tape (EOT). A reflective marker placed near the end of a tape and sensed during output. The marker signals that the tape is nearly full.

enter key. The key on the display terminal keyboard that, if pressed, tells the operating system to read the information you entered.

event control block (ECB). A control block used to record the status (occurred or not occurred) of an event; often used to synchronize the execution of tasks. ECBs are used in conjunction with the WAIT and POST instructions.

Event Driven Language (EDL). The language for input to the Event Driven Executive compiler ($EDXASM), or the Macro and Host assemblers in conjunction with the Event Driven Executive macro libraries. The output is interpreted by the Event Driven Executive emulator.

EXIO (execute input or output). An EDL facility that provides user controlled access to Series/1 input/output devices.

external label. A label attached to the outside of a tape that identifies the tape visually. It usually contains items of identification such as file name and number, creation data, number of volumes, department number, and so on.

external name (EXTRN). The 1- to 8-character symbolic EBCDIC name for an entry point or data field that is not defined within the module that references the name.

FCA. See file control area.

FCB. See file control block.

file. A set of related records treated as a logical unit. Although file is often used interchangeably with data set, it usually refers to an indexed or a sequential data set.

file control area (FCA). A Multiple Terminal Manager data area that describes a file access request.

file control block (FCB). The first block of an indexed file. It contains descriptive information about the data contained in the file.

file control block extension. The second block of an indexed file. It contains the file definition parameters used to define the file.

file manager. A collection of subroutines contained within the program manager of the Multiple Terminal Manager that provides common support for all disk data transfer operations as needed for transaction-oriented application programs. It supports indexed and direct files under the control of a single callable function.

floating point. A positive or negative number that can have a decimal point.

formatted screen image. A collection of display elements or display groups (such as operator prompts and field input names and areas) that are presented together at one time on a display device.

free pool. In an indexed data set, a group of blocks that can be used for either data blocks or index blocks. These differ from other free blocks in that these are not initially assigned to specific logical positions in the file.

free space. In an indexed file, records blocks that do not currently contain data, and are available for use.

free space entry (FSE). An 8-byte directory entry defining an area of free space within a volume or a device.

FSE. See free space entry.

general purpose interface bus. The IEEE Standard 488-1975 that allows various interconnected devices to be attached to the GPIB adapter (RPQ D02118).

Glossary of Terms and Abbreviations

GPIB. See general purpose interface bus.

group. A unit of 100 records in the spool data set allocated to a spool job.

H exchange format. A standard format for exchanging data on diskettes between systems or devices.

host assembler. The assembler licensed program that executes in a 370 (host) system and produces object output for the Series/1. The source input to the host assembler is coded in Event Driven Language or Series/1 assembler language. The host assembler refers to the System/370 Program Preparation Facility (5798-NNQ).

host system. Any system whose resources are used to perform services such as program preparation for a Series/1. It can be connected to a Series/1 by a communications link.

IACB. See indexed access control block.

IAR. See instruction address register.

ICB. See indexed access control block.

IIB. See interrupt information byte.

image store. The area in a 4978 that contains the character image table.

immediate data. A self-defining term used as the operand of an instruction. It consists of numbers, messages or values which are processed directly by the computer and which do not serve as addresses or pointers to other data in storage.

index. In an indexed file, an ordered collection of pairs of keys and pointers, used to sequence and locate records.

index block. In an indexed file, an area that contains control information and index entries. These blocks are a multiple of 256 bytes.

indexed access control block (IACB/ICB). The control block that relates an application program to an indexed file.

indexed access method. An access method for direct or sequential processing of fixed-length records by use of a record's key.

indexed data set. Synonym for indexed file.

indexed file. A file specifically created, formatted and used by the Indexed Access Method. An indexed file is sometimes called an indexed data set.

index entry. In an indexed file, a key-pointer pair, where the pointer is used to locate a lower-level index block or a data block.

index register (#1, #2). Two words defined in EDL and contained in the task control block for each task. They are used to contain data or for address computation.

input buffer. (1) See buffer. (2) In the Multiple Terminal Manager, an area for terminal input and output.

input output control block (IOCB). A control block containing information about a terminal such as the symbolic name, size and shape of screen, the size of the forms in a printer, or an optional reference to a user provided buffer.

instruction address register (IAR). The pointer that identifies the machine instruction currently being executed. The Series/1 maintains a hardware IAR to determine the Series/1 assembler instruction being executed. It is located in the level status block (LSB).

integer. A positive or negative number that has no decimal point.

interactive. The mode in which a program conducts a continuous dialogue between the user and the system.

internal label. An area on tape used to record identifying information (similar to the identifying information placed on an external label). Internal labels are checked by the system to ensure that the correct volume is mounted.

interrupt information byte (IIB). In the Multiple Terminal Manager, a word containing the status of a previous input/output request to or from a terminal.

invoke. To load and activate a program, utility, procedure, or subroutine into storage so it can run.

job. A collection of related program execution requests presented in the form of job control statements, identified to the jobstream processor by a JOB statement.

job control statement. A statement in a job that specifies requests for program execution, program parameters, data set definitions, sequence of execution, and, in general, describes the environment required to execute the program.

job stream processor. The job processing facility that reads job control statements and processes the requests made by these statements. The Event Driven Executive job stream processor is $JOBUTIL.

jumper. (1) A wire or pair of wires which are used for the arbitrary connection between two circuits or pins in an attachment card. (2) To connect wire(s) to an attachment card or to connect two circuits.

key. In the Indexed Access Method, one or more consecutive characters used to identify a record and establish its order with respect to other records. See also key field.

key field. A field, located in the same position in each record of an indexed file, whose content is used for the key of a record.

level status block (LSB). A Series/1 hardware data area that contains processor status. This area is eleven words in length.

library. A set of contiguous records within a volume. It contains a directory, data sets and/or available space.

line. A string of characters accepted by the system as a single input from a terminal; for example, all characters entered before the carriage return on the teletypewriter or the ENTER key on the display station is pressed.

link edit. The process of resolving external symbols in one or more object modules. A link edit is performed with $EDXLINK whose output is a loadable program.

listener. A controller or active device on a GPIB bus that is configured to accept information from the bus.

load mode. In the Indexed Access Method, the mode in which records are loaded into base record slots in an indexed file.

load module. A single module having cross references resolved and prepared for loading into storage for execution. The module is the output of the $UPDATE or $UPDATEH utility.

load point. (1) Address in the partition where a program is loaded. (2) A reflective marker placed near the beginning of a tape to indicate where the first record is written.

lock. In the Indexed Access Method, a method of indicating that a record or block is in use and is not available for another request.

logical screen. A screen defined by margin settings, such as the TOPM, BOTM, LEFTM and RIGHTM parameters of the TERMINAL or IOCB statement.

LSB. See level status block.

mapped storage. The processor storage that you defined on the SYSTEM statement during system generation.

member. A term used to identify a named portion of a partitioned data set (PDS). Sometimes member is also used as a synonym for a data set. See data set.

menu. A formatted screen image containing a list of options. The user selects an option to invoke a program.

menu-driven. The mode of processing in which input consists of the responses to prompting from an option menu.

message. In data communications, the data sent from one station to another in a single transmission. Stations communication with a series of exchanged messages.

multifile volume. A unit of recording media, such as tape reel or disk pack, that contains more than one data file.

multiple terminal manager. An Event Driven Executive licensed program that provides support for transaction-oriented applications on a Series/1. It provides the capability to define transactions and manage the programs that support those transactions. It also manages multiple terminals as needed to support these transactions.

multivolume file. A data file that, due to its size, requires more than one unit of recording media (such as tape reel or disk pack) to contain the entire file.

new high key. A key higher than any other key in an indexed file.

nonlabeled tapes. Tapes that do not contain identifying labels (as in standard labeled tapes) and contain only files separated by tapemarks.

null character. A user-defined character used to define the unprotected fields of a formatted screen.

option selection menu. A full screen display used by the Session Manager to point to other menus or system functions, one of which is to be selected by the operator. (See primary option menu and secondary option menu.)

output buffer. (1) See buffer. (2) In the Multiple Terminal Manager, an area used for screen output and to pass data to subsequent transaction programs.

overlay. The technique of reusing a single storage area allocated to a program during execution. The storage area can be reused by loading it with overlay programs that have been specified in the PROGRAM statement of the program or by calling overlay segments that have been specified in the OVERLAY statement of $EDXLINK.

overlay area. A storage area within a program reserved for overlay programs specified in the PROGRAM statement or overlay segments specified in the OVERLAY statement in $EDXLINK.

overlay program. A program in which certain control sections can use the same storage location at different times during execution. An overlay program can execute concurrently as an asynchronous task with other programs and is specified in the EDL PROGRAM statement in the main program.

overlay segment. A self-contained portion of a program that is called and sequentially executes as a synchronous task. The entire program that calls the overlay segment need not be maintained in storage while the overlay segment is executing. An overlay segment is specified in the OVERLAY statement of $EDXLINK or $XPSLINK (for initialization modules).

overlay segment area. A storage area within a program or supervisor reserved for overlay segments. An overlay segment area is specified with the OVLAREA statement of $EDXLINK.

parameter selection menu. A full screen display used by the Session Manager to indicate the parameters to be passed to a program.

partition. A contiguous fixed-sized area of storage. Each partition is a separate address space.

performance volume. A volume whose name is specified on the DISK definition statement so that its address is found during IPL, increasing system performance when a program accesses the volume.

physical timer. Synonym for timer (hardware).

polling. In data communications, the process by which a multipoint control station asks a tributary if it can receive messages.

precision. The number of words in storage needed to contain a value in an operation.

prefind. To locate the data sets or overlay programs to be used by a program and to store the necessary information so that the time required to load the prefound items is reduced.

primary file. An indexed file containing the data records and primary index.

primary file entry. For the Indexed Access Method Version 2, an entry in the directory describing a primary file.

primary index. The index portion of a primary file. This is used to access data records when the primary key is specified.

primary key. In an indexed file, the key used to uniquely identify a data record.

primary-level index block. In an indexed file, the lowest level index block. It contains the relative block numbers (RBNs) and high keys of several data blocks. See cluster.

primary menu. The program selection screen displayed by the Multiple Terminal Manager.

primary option menu. The first full screen display provided by the Session Manager.

primary station. In a Series/1-to-Series/1 Attachment, the processor that controls communication between the two computers. Contrast with secondary station.

primary task. The first task executed by the supervisor when a program is loaded into storage. It is identified by the PROGRAM statement.

priority. A combination of hardware interrupt level priority and a software ranking within a level. Both primary and secondary tasks will execute asynchronously within the system according to the priority assigned to them.

process mode. In the Indexed Access Method, the mode in which records can be retrieved, updated, inserted, or deleted.

processor status word (PSW). A 16-bit register used to (1) record error or exception conditions that may prevent further processing and (2) hold certain flags that aid in error recovery.

program. A disk- or diskette-resident collection of one or more tasks defined by a PROGRAM statement; the unit that is loaded into storage. (See primary task and secondary task.)

program header. The control block found at the beginning of a program that identifies the primary task, data sets, storage requirements and other resources required by a program.

program/storage manager. A component of the Multiple Terminal Manager that controls the execution and flow of application programs within a single program area and contains the support needed to allow multiple operations and sharing of the program area.

protected field. A field in which the operator cannot use the keyboard to enter, modify, or erase data.

PSW. See processor status word.

QCB. See queue control block.

QD. See queue descriptor.

QE. See queue element.

queue control block (QCB). A data area used to serialize access to resources that cannot be shared. See serially reusable resource.

queue descriptor (QD). A control block describing a queue built by the DEFINEQ instruction.

queue element (QE). An entry in the queue defined by the queue descriptor.

quiesce. To bring a device or a system to a halt by rejection of new requests for work.

quiesce protocol. A method of communication in one direction at a time. When sending node wants to receive, it releases the other node from its quiesced state.

record. (1) The smallest unit of direct access storage that can be accessed by an application program on a disk or diskette using READ and WRITE. Records are 256 bytes in length. (2) In the Indexed Access Method, the logical unit that is transferred between $IAM and the user's buffer. The length of the buffer is defined by the user. (3) In BSCAM communications, the portions of data transmitted in a message. Record length (and, therefore, message length) can be variable.

recovery. The use of backup data to re-create data that has been lost or damaged.

reflective marker. A small adhesive marker attached to the reverse (nonrecording) surface of a reel of magnetic tape. Normally, two reflective markers are used on each reel of tape. One indicates the beginning of the recording area on the tape (load point), and the other indicates the proximity to the end of the recording area (EOT) on the reel.

relative block address (RBA). The location of a block of data on a 4967 disk relative to the start of the device.

relative record number. An integer value identifying the position of a record in a data set relative to the beginning of the data set. The first record of a data set is record one, the second is record two, the third is record three.

relocation dictionary (RLD). The part of an object module or load module that is used to identify address and name constants that must be adjusted by the relocating loader.

remote management utility control block (RCB). A control block that provides information for the execution of remote management utility functions.

reorganize. The process of copying the data in an indexed file to another indexed file in a manner that rearranges the data for more optimum processing and free space distribution.

restart. Starting the spool facility w the spool data set contains jobs from a previous session. The jobs in the spool data set can be either deleted or printed when the spool facility is restarted.

return code. An indicator that reflects the results of the execution of an instruction or subroutine. The return code is usually placed in the task code word (at the beginning of the task control block).

roll screen. A display screen which is logically segmented into an optional history area and a work area. Output directed to the screen starts display at the beginning of the work area and continues on down in a line-by-line sequence. When the work area gets full, the operator presses ENTER/SEND and its contents are shifted into the optional history area and the work area itself is erased. Output now starts again at the beginning of the work area.

SBIOCB. See sensor based I/O control block.

second-level index block. In an indexed data set, the second-lowest level index block. It contains the addresses and high keys of several primary-level index blocks.

secondary file. See secondary index.

secondary index. For the Indexed Access Method Version 2, an indexed file used to access data records by their secondary keys. Sometimes called a secondary file.

secondary index entry. For the Indexed Access Method Version 2, this an an entry in the directory describing a secondary index.

secondary key. For the Indexed Access Method Version 2, the key used to uniquely identify a data record.

secondary option menu. In the Session Manager, the second in a series of predefined procedures grouped together in a hierarchical structure of menus. Secondary option menus provide a breakdown of the functions available under the session manager as specified on the primary option menu.

secondary task. Any task other than the primary task. A secondary task must be attached by a primary task or another secondary task.

secondary station. In a Series/1-to-Series/1 Attachment, the processor that is under the control of the primary station.

sector. The smallest addressable unit of storage on a disk or diskette. A sector on a 4962 or 4963 disk is equivalent to an Event Driven Executive record. On a 4964 or 4966 diskette, two sectors are equivalent to an Event Driven Executive record.

selection. In data communications, the process by which the multipoint control station asks a tributary station if it is ready to send messages.

self-defining term. A decimal, integer, or character that the computer treats as a decimal, integer, or character and not as an address or pointer to data in storage.

sensor based I/O control block (SBIOCB). A control block containing information related to sensor I/O operations.

sequential access. The processing of a data set in order of occurrence of the records in the data set. (1) In the Indexed Access Method, the processing of records in ascending collating sequence order of the keys. (2) When using READ/WRITE, the processing of records in ascending relative record number sequence.

serially reusable resource (SRR). A resource that can only be accessed by one task at a time. Serially reusable resources are usually managed via (1) a QCB and ENQ/DEQ statements or (2) an ECB and WAIT/POST statements.

service request. A device generated signal used to inform the GPIB controller that service is required by the issuing device.

session manager. A series of predefined procedures grouped together as a hierarchical structure of menus from which you select the utility functions, program preparation facilities, and language processors needed to prepare and execute application programs. The menus consist of a primary option menu that displays functional groupings and secondary option menus that display a breakdown of these functional groupings.

shared resource. A resource that can be used by more than one task at the same time.

shut down. See data set shut down.

source module/program. A collection of instructions and statements that constitute the input to a compiler or assembler. Statements may be created or modified using one of the text editing facilities.

spool job. The set of print records generated by a program (including any overlays) while engueued to a printer designated as a spool device.

spool session. An invocation and termination of the spool facility.

spooling. The reading of input data streams and the writing of output data streams on storage devices, concurrently with job execution, in a format convenient for later processing or output operations.

SRQ. See service request.

stand-alone dump. An image of processor storage written to a diskette.

stand-alone dump diskette. A diskette supplied by IBM or created by the $DASDI utility.

standard labels. Fixed length 80-character records on tape containing specific fields of information (a volume label identifying the tape volume, a header label preceding the data records, and a trailer label following the data records).

static screen. A display screen formatted with predetermined protected and unprotected areas. Areas defined as operator prompts or input field names are protected to prevent accidental overlay by input data. Areas defined as input areas are not protected and are usually filled in by an operator. The entire screen is treated as a page of information.

station. In BSCAM communications, a BSC line attached to the Series/1 and functioning in a point-to-point or multipoint connection. Also, any other terminal or processor with which the Series/1 communicates.

subroutine. A sequence of instructions that may be accessed from one or more points in a program.

supervisor. The component of the Event Driven Executive capable of controlling execution of both system and application programs.

system configuration. The process of defining devices and features attached to the Series/1.

SYSGEN. See system generation.

system generation. The processing of defining I/O devices and selecting software options to create a supervisor tailored to the needs of a specific Series/1 hardware configuration and application.

system partition. The partition that contains the root segment of the supervisor (partition number 1, address space 0).

talker. A controller or active device on a GPIB bus that is configured to be the source of information (the sender) on the bus.

tape device data block (TDB). A resident supervisor control block which describes a tape volume.

tapemark. A control character recorded on tape used to separate files.

task. The basic executable unit of work for the supervisor. Each task is assigned its own priority and processor time is allocated according to this priority. Tasks run independently of each other and compete for the system resources. The first task of a program is the primary task. All tasks attached by the primary task are secondary tasks.

task code word. The first two words (32 bits) of a task's TCB; used by the emulator to pass information from system to task regarding the outcome of various operations, such as event completion or arithmetic operations.

task control block (TCB). A control block that contains information for a task. The information consists of pointers, save areas, work areas, and indicators required by the supervisor for controlling execution of a task.

task supervisor. The portion of the Event Driven Executive that manages the dispatching and switching of tasks.

TCB. See task control block.

terminal. A physical device defined to the EDX system using the TERMINAL configuration statement. EDX terminals include directly attached IBM displays, printers and devices that communicate with the Series/1 in an asynchronous manner.

terminal control block (CCB). A control block that defines the device characteristics, provides temporary storage, and contains links to other system control blocks for a particular terminal.

terminal environment block (TEB). A control block that contains information on a terminal's attributes and the program manager operating under the Multiple Terminal Manager. It is used for processing requests between the terminal servers and the program manager.

terminal screen manager. The component of the Multiple Terminal Manager that controls the presentation of screens and communications between terminals and transaction programs.

terminal server. A group of programs that perform all the input/output and interrupt handling functions for terminal devices under control of the Multiple Terminal Manager.

terminal support. The support provided by EDX to manage and control terminals. See terminal.

timer. The timer features available with the Series/1 processors. Specifically, the 7840 Timer Feature card (4955 only) or the native timer (4952, 4954, and 4956). Only one or the other is supported by the Event Driven Executive.

trace range. A specified number of instruction addresses within which the flow of execution can be traced.

transaction oriented applications. Program execution driven by operator actions, such as responses to prompts from the system. Specifically, applications executed under control of the Multiple Terminal Manager.

transaction program. See transaction-oriented applications.

transaction selection menu. A Multiple Terminal Manager display screen (menu) offering the user a choice of functions, such as reading from a data file, displaying data on a terminal, or waiting for a response. Based upon the choice of option, the application program performs the requested processing operation.

tributary station. In BSCAM communications, the stations under the supervision of a control station in a multipoint connection. They respond to the control station's polling and selection.

unmapped storage. The processor storage in your processor that you did not define on the SYSTEM statement during system generation.

unprotected field. A field in which the operator can use the keyboard to enter, modify or erase data. Also called non-protected field.

update. (1) To alter the contents of storage or a data set. (2) To convert object modules, produced as the output of an assembly or compilation, or the output of the linkage editor, into a form that can be loaded into storage for program execution and to update the directory of the volume on which the loadable program is stored.

user exit. (1) Assembly language instructions included as part of an EDL program and invoked via the USER instruction. (2) A point in an IBM-supplied program where a user written routine can be given control.

variable. An area in storage, referred to by a label, that can contain any value during program execution.

vary offline. (1) To change the status of a device from online to offline. When a device is offline, no data set can be accessed on that device. (2) To place a disk or diskette in a state where it is unknown by the system.

vary online. To place a device in a state where it is available for use by the system.

vector. An ordered set or string of numbers.

volume. A disk, diskette, or tape subdivision defined using $INITDSK or $TAPEUT1.

volume descriptor entry (VDE). A resident supervisor control block that describes a volume on a disk or diskette.

volume label. A label that uniquely identifies a single unit of storage media.

Index

The following index contains entries for this book only. See the *Library Guide and Common Index* for a Common Index to all Event Driven Executive books.

Index

T

IBM Series/1 Event Driven Executive

Publications Order Form

Instructions:

1. Complete the order form, supplying all of the requested information. (Please print or type.)
2. If you are placing the order by phone, dial **1-800-IBM-2468.**
3. If you are mailing your order, fold the order form as indicated, seal with tape, and mail. We pay the postage.

Ship to:

Name:

Address:

City:

State: Zip:

Bill to:

Customer number:

Name:

Address:

City:

State: Zip:

Your Purchase Order No.:

Phone: ( )

Signature:

Date:

Order:

Description	Order number	Qty.
Reference books:		
Set of the following six books. To order individual copies, use the following order numbers.	**SBOF-1627**	
Communications Guide	SC34-0638	
Extended Address Mode and Performance Analyzer User Guide	SC34-0591	
Installation and System Generation Guide	SC34-0646	
Language Reference	SC34-0643	
Library Guide and Common Index	SC34-0645	
Messages and Codes	SC34-0636	
Operator Commands and Utilities Reference	SC34-0644	
Guides and reference cards:		
Set of the following four books and reference cards. To order individual copies, use the following order numbers.	**SBOF-1628**	
Customization Guide	SC34-0635	
Event Driven Language Programming Guide	SC34-0637	
Operation Guide	SC34-0642	
Problem Determination Guide	SC34-0639	
Language Reference Card	SX34-0165	
Operator Commands and Utilities Reference Card	SX34-0164	
Conversion Charts Reference Card	SX34-0163	
Reference Card Envelope	SX34-0166	
Set of three reference cards and storage envelope. (One set is included with order number SBOF-1627)	**SBOF-1629**	
Binders:		
3-ring easel binder with 1 inch rings	SR30-0324	
3-ring easel binder with 2 inch rings	SR30-0327	
Standard 3-ring binder with 1 inch rings	SR30-0329	
Standard 3-ring binder with 1 1/2 inch rings	SR30-0330	
Standard 3-ring binder with 2 inch rings	SR30-0331	
Diskette binder (Holds eight 8-inch diskettes.)	SB30-0479	

Publications Order Form

Cut or Fold Along Line

Fold and tape Please Do Not Staple Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
1 Culver Road
Dayton, New Jersey 08810

Fold and tape Please Do Not Staple Fold and tape

IBM®

International Business Machines Corporation

IBM Series/1 Event Driven Executive Extended Address Mode and Performance Analyzer User Guide

Order No. SC 34-0591-0

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.)

SC34-0591-0
Printed in U.S.A.

Cut or Fold Along Line

Reader's Comment Form

Fold and tape | Please Do Not Staple | Fold and tape

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Information Development, Department 28B
P.O. Box 1328
Boca Raton, Florida 33432

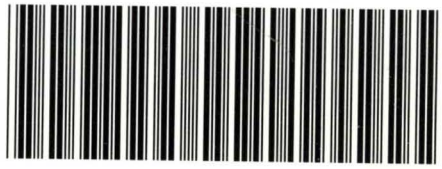
Fold and tape | Please Do Not Staple | Fold and tape

IBM®

IBM

International Business Machines Corporation

SC34-0591-0



SC34-0591-0
Program Number 5719-SX5
File No. S1-34
Printed in U.S.A.