LY28-1099-1
File No. S370-36

Program Product

OS/VS2
System Logic Library
Volume 11

Module Descriptions

MVS/System Product – JES3 5740-XYN
MVS/System Product – JES2 5740-XYS

Second Edition (December, 1982)

This is a major revision of, and obsoletes, LY28-1099-0 and Technical Newsletters LN28-5021 and LN28-4992. See the Summary of Amendments following the Contents for a summary of the changes made to this manual. Technical changes or additions to the text and illustrations are indicated by a vertical line to the left of the change.

This edition applies to Release 3.8 of OS/VS2 MVS with release 3 and to all subsequent releases of OS/VS2 until otherwise indicated in new editions or Technical Newsletters. Changes are made periodically to the information herein; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370 Bibliography*, GC20-0001, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM program product in this publication is not intended to state or imply that only IBM's program product may be used. Any functionally equivalent program may be used instead.

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

A form for reader's comments is provided at the back of this publication. If the form has been removed, comments may be addressed to IBM Corporation, Information Development, Department D58, Building 920-2, PO Box 390, Poughkeepsie, N.Y. 12602. IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Preface

The *System Logic Library* is intended for people who debug or modify the MVS control program. It describes the logic of most MVS control program functions that are performed after master scheduler initialization is complete. Refer to *OS/VS2 System Initialization Logic* for detailed information about the MVS control program prior to this point; to the *OS/VS2 MVS Overview* for general information about the MVS control program and the relationships among the components; and to the list of Corequisite Reading later in the preface to obtain the names of publications that describe components not included in the *System Logic Library.*

How the Book is Organized

The *System Logic Library* is comprised of eleven volumes. A copy of this preface, describing the organization and use of the book, can be found in each of the eleven volumes. Volume 1 contains the master table of contents and master index for the remaining volumes. Each of the other volumes contains its own index referring to the information in that particular volume; Volumes 2-10 each contain a table of contents. Volumes 2-10 describe the logic of the components in the MVS control program. Volume 11 contains module descriptions for all of the modules in the components documented in the *System Logic Library.*

The information in Volumes 2 through 10 is organized alphabetically by section name. Each section contains information pertaining to one or more of the components in the MVS control program. A section contains more than one component when the components are closely related, frequently referenced at the same time, and not so large that they require a volume of their own.

A three or four character mnemonic is associated with each section and used in the identification of all figures, diagrams, and pages in that section. For example, the mnemonic ASM is associated with the section "Auxiliary Storage Management." All figures in this section are identified as Figure ASM-n, all diagrams as Diagram ASM-n, and all pages as ASM-n, where n represents the specific figure, diagram, or page number. Whenever possible, existing component acronyms are used as the mnemonic for a section. The mnemonics have all been chosen so that they are in alphabetic order along with the full section name. The section names, the components included in each section (if a section contains more than one component), the mnemonics for the sections, and the volume and order number for each section are given in the table on page v.

The order number for Volume 1 (containing the master table of contents and index) is SY28-0713; the order number for Volume 11 (containing the module descriptions) is LY28-1099. Please note that the order numbers are not consecutive. This has been done to allow for the expansion of existing sections and the addition of the new sections.

How to Use the Book

In order to use this book efficiently, readers must be able to find the information that they need quickly; they must be aware of the types of information provided for each component; and they must know how to obtain additional information before referencing the *System Logic Library*. The following topics cover these points.

Finding Information Using the Master Index

Readers who are not familiar with the organization of the *System Logic Library* can locate information by using the master index in Volume 1. All index entries in volume 1 specify a volume number in parenthesis, followed by a page number. For Volumes 2-10, the page number consists of the mnemonic for the section and the page number; for Volume 11 (which contains the module descriptions), the page number consists of the volume number instead of a mnemonic and the page number. For example:

(3) ASM-12	refers to Volume 3, the "Auxiliary Storage Management" section, page ASM-12.
(11) 11-15	refers to Volume 11, page 11-15.

The volume number and section mnemonic are not repeated for successive references to the same section in a single entry in the master index; for example, (3) ASM-12, 17 refers to both pages 12 and 17. The volume number in parenthesis is not included in the individual index entries in Volumes 2-11, since all entries apply to the same volume.

Finding Information Using the Volume Titles

As readers become familiar with the section names, their mnemonics, and contents, they will not need to go to Volume 1, but will be able to use the individual indexes in each volume. They will be able to use the *System Logic Library* as they would an encyclopedia and go directly to the volume that they need. To help readers locate the correct volume, section mnemonics are included in the titles of Volume 2-10. If a volume contains one section, the mnemonic for that section is specified; if a volume contains more than one section, the mnemonics for the first and last section in the volume are specified. For example:

Volume 2 (ALC)	indicates that Volume 2 contains only the section identified by the mnemonic ALC (Allocation/Unallocation).
Volume 6 (CSV-PCA)	indicates that Volume 6 contains the sections identified by the mnemonics CSV and PCA as well as all the sections that fall between them in alphabetical order. Therefore, volume 6 contains the sections identified by the mnemonics CSV (Contents Supervision), CVI (Converter/Interpreter), ENF (Event Notification Facility), INIT (Initiator/ Terminator), MSI (Master Subsystem/Subsystem interface), and PCA (PC/AUTH Service Routines).

The list of section names and mnemonics on page v of this preface is included in every volume and provides a quick reference to the mnemonics and the components included in each section.

Table of Section Names

Section Name	Section Mnemonic	Volume and Order Number
Allocation/Unallocation	ALC	Volume 2 LY28-1063
Auxiliary Storage Management Checkpoint/Restart	ASM CHK	Volume 3 LY28-1067
Command Processing Includes: Command Processing System Log Region Control Task System Component Address Space Initialization Started Task Control	CMD	Volume 4 LY28-1071
Communications Task	COM	Volume 5 LY28-1075
Contents Supervision (Program Management) Converter/Interpreter Event Notification Facility Initiator/Terminator Master Subsystem/Subsystem Interface: Includes: Master Subsystem Subsystem Interface SWA Create Interface SWA Manager PC/AUTH Service Routines	CSV CVI ENF INIT MSI PCA	Volume 6 LY28-1079
Real Storage Management Recovery Support (Recovery Management Support) Includes: Machine Check Handler Power Warning Feature Channel Check Handler Dynamic Device Reconfiguration Missing Interrupt Handler	RSM RSP	Volume 7 LY28-1083
Recovery Termination Management Includes: RTM1 RTM2 Dumping Services SLIP	RTM	Volume 8 LY28-1087
Scheduler JCL Facility Supervisor Control System Management Facilities	SCH SCTL SMF	Volume 9 LY28-1091
System Resources Manager Task Management Timer Supervision Virtual Storage Management	SRM TASK TIME VSM	Volume 10 LY28-1095
Note: TSO LOGON has been moved to *OS/VS2 TSO Command Processor Logic, Volume IV.*		

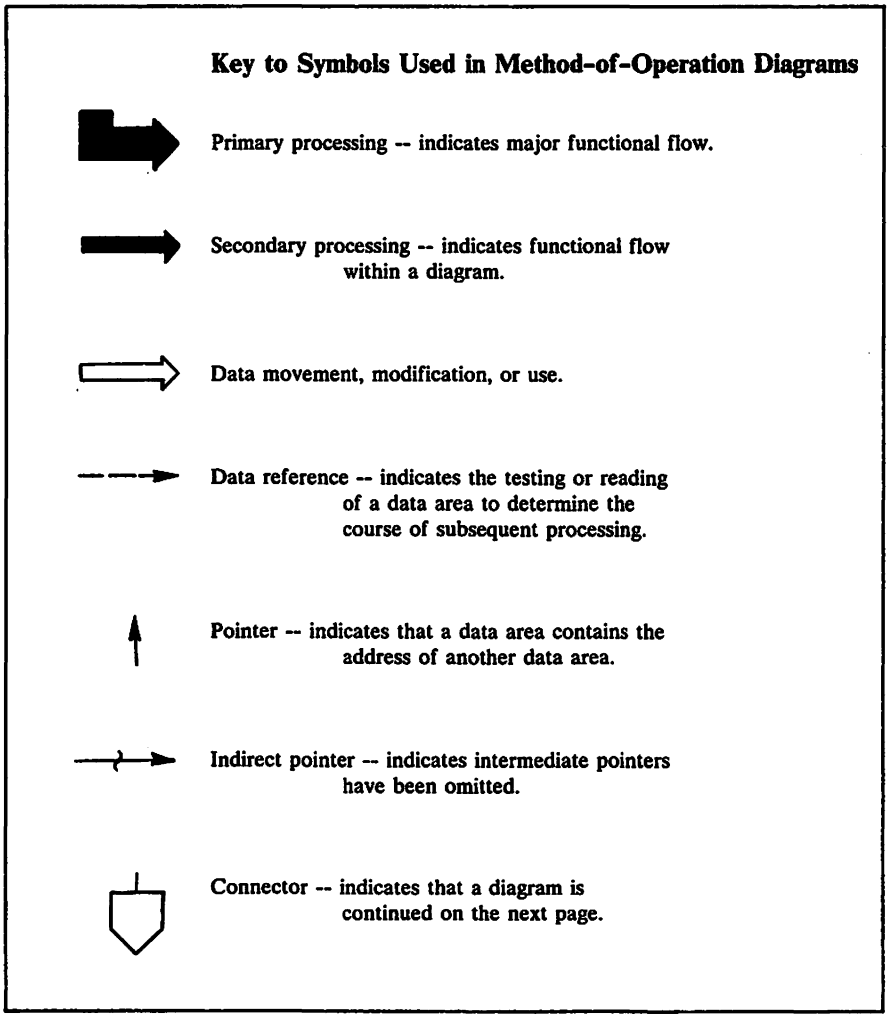
Information Provided for Each Component

The following information is provided for each of the components described in the *System Logic Library*.

1. An introduction that summarizes the component's function
2. Module flow diagrams that show control flow between the component's object modules
3. Control block overview diagrams that show significant fields and the chaining structure of the component's control blocks
4. Method-of-operation diagrams (sometimes called HIPOs or logic diagrams) that describe the functional organization of a program
5. Module descriptions that describe the operation of the modules

Items 1 through 4 are located in Volumes 2-10 in the section pertaining to the component; item 5 is located in Volume 11.

The following figure shows the symbols used in method-of-operation diagrams. The relative size and the order of fields in control block illustrations do not always represent the actual size and format of the control block.



Paths Into the System Logic Library

As a system programmer you might need additional information before referring to the *System Logic Library*. Other books in the library support debugging; understanding the information they provide can help you use the *System Logic Library* more efficiently. Possible problems and where to find helpful information follows. The exact titles of the microfiche publications are provided under Corequisite Reading later in the preface.

- If you need to know the detailed structure of a data area, refer to *Data Areas* (microfiche) or to Volumes 2 and 3 of the *OS/VS2 System Programming Library: Debugging Handbook*. You will find miscellaneous debugging information in Volume 1 of this publication.
- If you are trying to determine which modules use or change a data area or field, refer to the *Data Areas Usage Table* (microfiche) or to the *Symbol Usage Table* (microfiche).
- If you need to correlate CSECT names, aliases, and entry points of a load module, refer to the *Directory* (microfiche).
- To determine what module issues a message or detects the error conditions documented in the message, refer to the appendix of *OS/VS Message Library: VS2 System Messages.*
- To determine the meaning of an abend code and the module that detects or issues it, refer to *OS/VS Message Library: VS2 System System Codes.*
- If you have not identified the failing component, refer to *OS/VS2 System Programming Library: Diagnostic Techniques*, which documents a debugging methodology, including information on analyzing stand-alone dumps. *Diagnostic Techniques* also contains diagnostic information for some components.

Corequisite Reading

Prerequisite information can be found in:

- *OS/VS2 MVS Overview*, GC28-0984

Other publications that support the task of debugging are:

- For MVS/SP-JES2 Release 2:
  - *Data Areas*, LYB8-1051 (microfiche)
  - *Data Areas Usage Table*, LYB8-1052 (microfiche)
  - *Directory*, LYB8-1053 (microfiche)
  - *Symbol Usage Table*, LYB8-1054 (microfiche)
- For MVS/SP-JES3 Release 2:
  - *Data Areas*, LYB8-1055 (microfiche)
  - *Data Areas Usage Table*, LYB8-1056 (microfiche)
  - *Directory*, LYB8-1057 (microfiche)
  - *JES3 Messages*, GC23-0044
  - *Symbol Usage Table*, LYB8-1058 (microfiche)
- *OS/VS Message Library: VS2 System Codes*, GC38-1008
- *OS/VS Message Library: VS2 System Messages*, GC38-1002
- *OS/VS2 System Programming Library: MVS Diagnostic Techniques*, GC28-0725
- *OS/VS2 System Programming Library: Debugging Handbook*, Volumes 1, 2, and 3, GC28-1047, GC28-1048, and GC28-1049

The following publications provide logic information for components not described in the *System Logic Library*:

- Global resource serialization - *OS/VS2 Global Resource Serialization Logic*, LY28-1059
- I/O supervisor - *OS/VS2 I/O Supervisor Logic*, SY26-3823
- JES2 - *JES2 Logic*, LY24-6006
- JES3 - *JES3 Logic*, LY24-6005
- RMF - *OS/VS2 MVS Resource Measurement Facility (RMF) Program Logic Manual*, LY28-0923
- System initialization - *OS/VS2 System Initialization Logic*, LY28-1050
- TCAM - *OS/VS2 TCAM Level 10 Logic*, SY30-3032
- TSO LOGON - *OS/VS2 TSO Command Processor Logic Volume IV*, SY28-0652
- VTAM - *OS/VS2 MVS VTAM Logic*, SY28-0621

The following publications are referenced for specific components:

- Checkpoint/restart - *OS/VS2 MVS Checkpoint/Restart*, GC26-3877
- Dumping services - *OS/VS2 MVS System Programming Library: Service Aids*, GC28-0674
- SMF - *OS/VS System Management Facilities*, GC28-1030
- SRM - *OS/VS2 System Programming Library: Initialization and Tuning Guide*, GC28-1029
- Virtual Fetch - *OS/VS2 System Programming Library: Job Management*, GC28-0627

Note: These titles are sometimes used in abbreviated form in the text.

Summary of Amendments

Summary of Amendments
for LY28-1099-1
MVS/System Product Release 3.3

This edition incorporates changes to the module descriptions in support of MVS/System Product Release 3.3.

Summary of Amendments
for LY28-1099-0
as Updated June 30, 1982
by Technical Newsletter LN28-5021

Module descriptions of new and changed modules for MVS System Product Release 3.2 have been updated.

Summary of Amendments
for LY28-1099-0
as Updated December 30, 1981
by Technical Newsletter LN28-4992

This newsletter contains an updated module description of ILRPREAD.

Module Descriptions

This section contains descriptions of the object modules that perform the scheduler and supervisor functions for the MVS operating system. The description of each object module includes:

- A summary of the operation of the module
- The names of the modules that pass control to it
- The names of the modules that receive control from it

The module descriptions are arranged alphamerically by module name. If there are two names in a title, the first is the CSECT name and the second (in parentheses) is the assembler module name. If there is only one name in the title, it is the name of both the CSECT and the assembler module.

CSVVFCRE — Virtual Fetch Initialization Module

Operation: CSVVFCRE builds the virtual fetch control block (VFCB) in the common service area (CSA) and sets up a cross memory environment for the virtual fetch service address space. CSVVFCRE then builds the virtual fetch hash table and the virtual fetch VIO data set. Using the data sets provided by the user, virtual fetch reformats and validates the directory entries (DEs) before placing them in the virtual fetch hash table. CSVVFCRE places the user-provided load modules into the VIO data set. When finished, each entry in the hash table contains the location of a load module in the VIO data set. CSVVFCRE is a never-ending task. CSVVFCRE enters a wait state until the operator either reexecutes CSVVFCRE (via a cross memory post from CSVVFRSH) or cancels CSVVFCRE.

Entered through: Started task control.

Exit to: None.

Called routines: CSVVFCR1, IEAVAMSI, ILRGOS.

CSVVFCR1 — Virtual Fetch Module Formatter

Operation: CSVVFCRE calls CSVVFCR1 to reformat a load module that CSVVFCRE provides. CSVVFCR1 reads the TXT records into a buffer. CSVVFCR1 then reformats the TXT and RLD data and copies this data into the VIO window. CSVVFCR1 converts the directory entry (DE) for this module into the virtual fetch hash table format and places the reformatted DE in the hash table. Since CSVVFCRE can access the window, CSVVFCR1 only returns a return code stating whether the reformatting was successful.

Entry from: CSVVFCRE.

Exit to: Caller.

CSVVFGET — Virtual Fetch Module Obtain Routine

Operation: Callers use CSVVFGET to bring a module into storage from virtual fetch's read-only VIO data set and to pass control to the requested

module. CSVVFGET validates the caller's input, issues a call to CSVVFTCH to bring the module into storage, issues a SYNCH SVC to the module, and cleans up.

Entry from:

For entry point CSVVFGET: Users of virtual fetch.

For entry point CSVVFRM: CSVVFIND, CSVVFTCH (to issue the FREEMAINs for module and VCBs storage).

For entry point CSVVFGES: RTM.

For entry point CSVVFRR: RTM.

Exit to: Caller.

Error exit: RTM

Called routines: CSVVFIND, CSVVFTCH.

CSVVFIND — Virtual Fetch BUILD and FIND Routine

Operation: Callers use CSVVFIND to build a VFWK to represent a module accessible by virtual fetch. Callers also use CSVVFIND to determine if a VFWK exists and, if necessary, to initialize the VFWK. CSVVFIND validates the parameter list, performs initialization, and searches for the VFWK. If the caller requested the BUILD function and the VFWK already exists, CSVVFIND returns to the caller; if the VFWK does not exist, CSVVFIND creates one, initializes it, and chains it to the VFWK chain. If the caller requested the FIND function, CSVVFIND finds the VFWK and checks to see if the VFDE data in the VFWK needs to be refreshed. (CSVVFIND calls CSVVFSCH to perform the VFWK refresh operation.)

Entry from:

For entry point CSVVFIND: User of the virtual fetch service.

For entry point CSVVFORK: CSVVFGET, CSVVFIND.

For entry point CSVVFFES: RTM.

Exit to: Caller.

Error exit: RTM.

Called routines: CSVVFSCH, CSVVFGET.

CSVVFMEM — Virtual Fetch Memory Termination Manager

Operation: During termination of the virtual fetch service address space, CSVVFMEM indicates in the VFCB that virtual fetch is not active. During termination of job step task, CSVVFMEM clears the pointer in the ASXB (ASXBVFVT) to the virtual fetch control blocks, located in the user's address space.

Entry from: IEAVTMMT, IEAVTSKT.

Exit to: Caller.

CSVVFRSH — Virtual Fetch Refresh Module

Operation: CSVVFRSH issues a cross memory post to CSVVFCRE to refresh the modules maintained by virtual fetch. If unable to issue the cross memory post (because a refresh is currently executing or virtual fetch is inactive), CSVVFRSH issues a WTO SVC (SVC 35) to inform the operator.

Entry from: Dispatcher.

Exit to: Caller.

CSVVFSCH — Virtual Fetch Search Routine

Note: CSVVFSCH and CSVVFCRE must be link edited into the same load module so that CSVVFCRE is able to set up a cross memory entry table containing CSVVFSCH.

Operation: CSVVFIND calls CSVVFSCH when it needs to refresh the VFDE data in a virtual fetch entry point work area (VFWK) in the user's address space. (The VFWK is the local control block for a module maintained by virtual fetch.) CSVVFSCH searches the virtual fetch hash table for the virtual fetch directory entry (VFDE) for the requested module. If found, CSVVFSCH returns the VFDE data to CSVVFIND (in the user's address space). CSVVFIND needs to call CSVVFSCH the first time it uses the FIND function and whenever the copy of the module located by the VFWK is obsolete and needs to be refreshed.

Entry from: CSVVFIND.

Exit to: Caller.

CSVVFTCH — Module-obtain Routine for Virtual Fetch

Operation: CSVVFGET calls CSVVFTCH to obtain a copy of a module. CSVVFTCH obtains sufficient storage for the module and the VIO control blocks (VCBs), initializes the CDE and extent list (XTLST) for the module, loads a fresh copy of the module from virtual fetch's VIO data set, and relocates the module's address constants (ADCONs).

Entry from: CSVVFGET.

Exit to: Caller

Called routines: IEAVAMSI.

ICFBDF00 — PWF Machine Check Handler Appendage

Operation: This module determines the severity of a power disturbance. If the power supply is in jeopardy, it transfers real storage to a disk.

Entry from: IGFPMCIH at entry point ICFDBE00.

Secondary entry point: ICFBDE99. This entry point is returned to by the user-provided code when a dump of real storage is warranted.

Exit to: Disabled wait state if the power warning is serious; IGFPMCIH if the power warning is false, if the Power Warning Feature is not supported, or if requested by the user-provided code.

IEAFTEED — EED Control Block Format Module

Operation: This module formats the EED passed in the parameter list and summarizes the bit flags. It sends back a return code to its caller indicating failure or success in formatting.

Entry from: IEAVTFMT.

Exit to: Caller.

IEAFTESA — ALIAS Entry in IEAFTRT2

IEAFTFRR — ALIAS Entry in IEAFTIHS

IEAFTIHS — IHSA-FRRS Control Block Format Module

Operation: This module formats the IHSA or FRRS passed in the parameter list. It also formats the RT1W which is embedded in both the IHSA and FRRS. It then summarizes the bit flags from the RT1W and passes back the address of the EED pointed to by the RT1W. It also sends back a return code to its caller indicating failure or success in formatting.

Entry from: IEAVTFMT at all entry points.

Exit to: Caller.

IEAFTRTC — ALIAS Entry in IEAFTSCB

IEAFTRT2 — RTM2WA-ESA-SDWA Format Module

Operation: This module formats the RTM2WA passed in the parameter list, summarizes the important fields of the RTM2WA, and summarizes the bit flags. It also summarizes the bit flags from the ESA of the related SVRB and the registers at the time of error from the SDWA. The RTM2WA entry point also returns the address of the related SVRB to the caller. Each entry point invoked sends a return code to its caller indicating success or failure in formatting.

Entry from: IEAVTFMT at all entry points.

Exit to: Caller.

IEAFTSCB — SCB-RTCT Control Block Format Module

Operation: This module formats the SCB or RTCT passed in the parameter list. It summarizes the bit flags in the SCB. It also sends a return code to its caller indicating success or failure.

Entry from: IEAVTFMT at all entry points.

Exit to: Caller.

IEAFTSDW — ALIAS Entry in IEAFTRT2

IEASMFEX — SMF EXCP Counter Routine

Operation: This module is entered to count all EXCPs into the TCTIOT and the OUCB.

Entry from: I/O drivers.

Exit to: Caller.

IEASMFSP — SMF Suspend/Reset Interface Module

Operation: This module provides SMF with an interface to the suspend/reset functions residing in the nucleus. The entry code indicates which function is requested.

Entry from: IEEMB830 (suspend), IEEMB825 (reset), IEEMB834 (reset).

Exit to: Dispatcher, for suspend. Caller, for reset.

Called routines: IEAVSUSQ, IEAVRSET (entry points in IEAVEPC, the PCFLIH).

IEATLEXT — SMF Time Limit Extension

Operation: This module is entered if a job, job step, or wait-time limit has expired. It interfaces with the SMF user's exit routine, IEFUTL. If the exit grants an extension, IEATLEXT extends the expired limit by the amount specified by the exit. If the exit does not grant an extension, IEATLEXT will abend the expired task with a code 322 for a job or job step limit expiration, or 522 for a wait-time limit expiration.

Entry from: Dispatcher (IEAVEDS0).

Exit to: Dispatcher (IEAVEDS0).

IEAV1052 — 1052 Console Device Support Processor

Operation: This module performs the following functions for a 1052 printer-keyboard, 3210 console printer-keyboard, 3215 console printer-keyboard, or 3213 console printer used as a console:

- Opens the device as a console.
- Initiates a read operation to read a command from the console.
- After a read operation completes successfully, passes the command to command processing (SVC 34).
- Initiates a write operation to write messages to the console.
- After a write operation completes successfully, updates the console queue.
- When the device is to be removed from console status, closes the console.

Entry from: IEAVVCTR.

Exit to:

- IEAVSWCH if an I/O error occurs or the console cannot be opened.
- Issuer of SVC 72, otherwise.

IEAV1443 — 1443 Console Device Support Processor

Operation: This module performs the following functions for a 1443 printer, 1403 printer, or 3211 printer used either as the output portion of a composite console or as an output-only console:

- Opens the printer as an output console.
- Uses BSAM to write WTO and WTOR messages to the console.
- After a write operation completes successfully, updates the console.
- When the printer is to be removed from console status, closes the console.

Entry from: IEAVVCTR.

Exit to:

- IEAVSWCH if an I/O error occurs or the console cannot be opened.
- Issuer of SVC 72, otherwise.

IEAV2540 — 2540 Console Device Support Processor

Operation:This module performs the following functions for a 2540 card reader punch, 2501 card reader, 2520 card reader punch, 3505 card reader, or 3525 card punch used as the input portion of a composite console:

- Opens the device as an input console.
- Uses BSAM to read a command from the console.
- After a read operation completes successfully, passes the command to command processing (SVC 34).
- When the device is to be removed from console status, closes the console.

Entry from: IEAVVCTR.

Exit to:

- IEAVSWCH if an I/O error occurs or the console cannot be opened.
- Issuer of SVC 72, otherwise.

IEAVADFM — User Formatter CSECT

Operation: This module contains no executable code, only installation formatter names.

Entry from: Not called.

Exit to: Not applicable.

IEAVAD0A — Display SQA and LSQA

Operation: This module displays the allocated SQA and LSQA.

Entry from: IEAVAD01 via BALR.

Exit to: IEAVAD01.

IEAVAD0B — Display LPA, JPA, and Active SVC Modules

Operation: This module conditionally displays those LPA/JPA modules that are on the active RB queue or load list of the task being dumped, and the active SVC modules for the task.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD0C — Supervisor Trace Formatting

Operation: This module displays the supervisor trace table.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD0D — Display Allocated Space Within User Subpools

Operation: This module displays allocated space in user subpools subpool 252.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD0E — Display SWA

Operation: This module displays allocated space in the SWA. This module is also called to display subpools 229 and 230 when LSQA is requested.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD0F — Display SNAP

Operation: This module displays the storage represented by the list supplied by the caller. This module also adds headers to the storage list if supplied by the user.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD00 — SVC 51

Operation: This module identifies the function requested (SNAP or SVC DUMP) and routes control to the corresponding routine. The SVC DUMP ROUTINE is contained in the module.

Entry from: SVC IH.

Exit to: Exit routine.

Exit-error: Abend caller.

IEAVAD01 — SNAP Mainline

Operation: This module gets storage for, and initializes, buffers, and validity checks SNAP parameter list. It also routes control to the appropriate functional display routine, displays an index of the dump, and cleans up after all indicated areas have been displayed.

Entry from: IEAVAD00 via BALR.

Exit to: IEAVAD00.

IEAVAD02 — Header, PSW Formatting

Operation: This module formats and prints the dump ID, jobname, stepname, time, and completion code.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD03 — Control Blocks I

Operation: This module formats and prints the ASCB, TCB, RBS, LLES, CDES, extent lists, and the TIOT for the task being dumped.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD05 — Control Blocks II

Operation: This module formats and displays the VMM blocks (SPQE, DQE, and FQE), the IQE, PQE, FBQE queues, and the TCB summary.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD07 — Display Save Areas

Operation: This module provides the display of user save areas.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD08 — SNAP, TCAM, GTF, VSAM, Shared Resources Interface Module, and Installation Formatters

Operation: This module is responsible for the interfaces to the SNAP, TCAM, GTF, VSAM, shared resources formatting routines, and installation format routines.

Entry from: IEAVAD01 via BALR.

Exit to: IEAVAD01, IBGDSNAP via BALR for GRSQ, IDA0195A via BALR for VSAM, IGA0E05A via LOAD and BALR for TCAM, IGC0F05A via LOAD and BALR for GTF, IGC0905A via LOAD and BALR for shared resources. ISTRAFD1 via LOAD and BALR for VTAM, and installation format routines whose names appear in the format CSECT, IEAVADFM via BALR.

IEAVAD09 — Display Nucleus and PSA

Operation: This module displays the control program nucleus and PSA.

Entry from: IEAVD01 via BALR.

Exit to: Caller.

IEAVAD10 — Display PSW and Registers

Operation: This module conditionally formats and prints the PSW, ILC, and interruption code, and displays the registers at entry to SNAP or ABEND.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD11 — SNAP Output Routines

Operation: This module writes the lines of the dump on the output device.

Entry point IEAVAD21 forces output of any records remaining in buffer after the dump is complete.

Entry point IEAVAD81 is the interface between the subsystem formatting routines and the output module IEAVAD11.

Entry from: IEAVAD01 via BALR.

Exit to: Caller.

IEAVAD31 — SNAP Format and Format 01 Routines

Operation: This module unpacks data into the output line while it associates a label with each piece of data.

Entry point IEAVAD31 converts the data to decimal notation and unpacks it into the output line associating labels with the fields. It also allows an indentation factor to the location in an output line where the data is to be placed.

Entry point IEAVAD41 performs the same functions as IEAVAD31 except that no indentation factor is anticipated in the layout line.

Entry from: Caller via BALR.

Exit to: Caller.

Error exit: RTM via on x'0C4' abend after an unexpected UPR occurs.

IEAVAD51 — SNAP Format 20 and Format 22 Routines

Operation: Format 20 translates the line into printable characters and unpacks three bytes of data into the beginning of the line. Format 22 unpacks up to four bytes of data into any positions of the line.

Entry from:

For entry point IEAVAD51 — format 20 routine: IEAVAD71 via BALR.

For entry point IEAVAD61 — format 22 routine: IEAVAD71 via BALR.

Exit to: IEAVAD71.

Error exit: IEAVAD71 with register 15 set to 8.

IEAVAD71 — SNAP Format Router

Operation: This routine controls the routing of other format routines to produce the print line that consists of the address of the data, the hex representation, and the EBCDIC translation of the data at that location.

Entry from: IEAVAD03, IEAVAD0B, IEAVAD0D via BALR.

Exit to: Caller.

Error exit: Caller with a return code of eight if the space for a save area is not available, or RTM with an X'0C4' abend if an unexpected UPR occurs.

IEAVAMSI — VIO Services Interface

Operation: This module manipulates page and external page tables and page frame table entries (PFTEs) for the virtual block processor (VBP) in support of VIO data sets.

Entry from: The virtual block processor (VBP).

Exit to: VBP.

IEAVAR00 — RCT Initialization/Termination

Operation: This module performs address space initialization functions for a new address space and attaches the dump task and started task control. When a task terminates, this module frees address space resources before termination and detaches the dump task and the started task control task.

Entry from:

For entry point IEAVAR00: IEAVEMIN.

For entry point IEAVAER0: IEAVTAS1.

Exit to: Caller.

IEAVAR01 — RCT Common Processing

Operation: This modules routes control within the RCT modules to wait for a functional request in the quiesce module and passes control to other routines to perform the requested functions. These functions are performed until a termination request is received, this module then returns control.

Entry from:

For entry point IEAVAR01: IEAVAR00.

For entry point IEAVARIA: IEAVTRT2.

Exit to: Caller.

IEAVAR02 — Quiesce Routine

Operation: This module waits for a functional request to be posted and determines the function requested. If posted by SRM, this module either prepares an address for swap-out, or verifies that all tasks under the RCT are in long wait prior to returning. If a swap-out is possible, this module breaks active addressing binds to the address space and prohibits new ones. If posted for a termination or attention exit request, this module returns control.

Entry from:

For entry point IEAVAR02: IEAVAR01.

For entry point IEAVAFR2: IEAVTRTS.

Exit to: Caller.

Called routines: IEAVEBBR, IEAVESPM, IEAVSOUT, IEAVWAIT.

IEAVAR03 — Restore Routine

Operation: This module either prepares an address space for execution after swap-in or handles quiesce backout processing.

Entry from:

For entry point IEAVAR03: IEAVAR01.

For entry point IEAVAR3A: IEAVTAS1.

For entry point IEAVAFR3: IEAVTRTS.

Exit to:

For entry points IEAVAR03 and IEAVAR3A: IEAVAR01.

For entry point IEAVAFR3: IEAVTRTS.

IEAVAR04 — Attention Exit Scheduler Routine

Operation: This module determines the attention level to be scheduled and schedules the associated attention exit.

Entry from:

For entry point IEAVAR04: IEAVAR01.

For entry point IEAVAFR4: IEAVTRTS.

Exit to: Caller.

IEAVAR05 — Attention Exit Prolog Routine

Operation: This module acts as a front end of the user attention exit. It issues TPUT or TGET to the terminal, creates and initializes the TAIE, and goes to the attention exit in the user's key and state.

Entry from:

For entry point IEAVAR05: IEAVEDS0.

For entry point IEAVART5: IEAVTRT2.

For entry point IEAVAFR5: IEAVTRTS.

Exit to:

For entry points IEAVAR05 and IEAVART5: User attention exit or IEAVEOR.

For entry point IEAVAFR5: Caller.

IEAVAR06 — Attention Exit Epilog Routine

Operation: This module acts as a resource manager for the TAXE queue. It frees the TAIE, restarts the subtasks of the TCB for which the attention exit has completed, performs housekeeping on the TAXE queue, and updates the available TAXE count.

Entry from:

For entry point IEAVAR06: IEAVEOR.

For entry point IEAVAFR6: IEAVTRTS.

Exit to: Caller.

IEAVAR07 — Attention Exit Purge Routine

Operation: This module acts as a resource manager for the TAXE queue. It is entered at task termination time so that inactive TAXEs for the terminating TCB can be dequeued. In addition to the TAXE queue housekeeping, this module updates the available TAXE count.

Entry from:

For entry point IEAVAR07: IEAVTSKT.

For entry point IEAVAFR7: IEAVTRTS.

Exit to: Caller.

IEAVAX00 — STAX SVC Service Routine

Operation: This routine creates, maintains, and releases terminal attention exit elements (TAXEs) which contain information for scheduling terminal attention exits. These TAXEs are queued from the Region Control Task Data Area (RCTD) as they are created.

Entry from:

For entry point IGC0009F: IEAVESVC.

For entry point STXFRR: IEAVTRTS.

Exit to:

For entry point IGC0009F: IEAVEOR.

For entry point STXFRR: Caller.

IEAVBLDP — Build Quickcell Pool

Operation: This module creates, extends, or recovers (reformats) a quick-cell pool.

Entry from: IEABLDP.

Exit to: Caller.

IEAVCARR — Functional Recovery Routine for Module IEAVGCAS

Operation: This module attempts to clean up and recover from the following error situations encountered in module IEAVGCAS: program checks, machine checks, abends, percolation from a subordinate FRR, and operator restarts.

Entry from:

For entry point IEAVFCARR, IEAVTTRR, and IEAVFARR: IEAVTRTS.

Exit to: Caller.

IEAVCKEY — Change Storage Key

Operation: This module changes the storage key and fetch protection for virtual pages allocated from the problem program subpools.

Entry from: A supervisor state routine executing under protection key zero by branch entry only.

Exit to: Caller.

IEAVCKRR — Functional Recovery Routine for module IEAVCKEY

Operation: This module attempts to recover from program checks and machine checks encountered in module IEAVCKEY by restoring the original storage key of those pages changed.

Entry from: RTM routine IEAVTRTS.

Exit to: Caller.

IEAVCSEG — Create Segment

Operation: This module performs one of the following functions for one or more segments at a time:

- Initializes a segment table entry and its associated page table

- Initializes a segment table entry and its associated page table, the external page table associated with the page table, and an entry in the SWAP control table (SPCT) for this segment

Entry from:

For entry point IEAVCSEG: IEAVNP08 and IEAVGM00.

For entry point IEAVCSGB: IEAVEQR.

Exit to: Caller.

IEAVC700 — Communications Task Command Queuer

Operation: This module queues a command for execution by the communications task, ensuring that the command executes in the communications task's address space and under its TCB. After the command is processed, this module dequeues the command.

Entry from: IEAVMQWR, IEE6703D.

Exit to: Caller.

IEAVDELP — Delete Quickcell Pool

Operation: This module deletes all or part of a quickcell pool.

Entry from: IEAVDELP.

Exit to: Caller.

IEAVDLAS — Delete Address Space

Operation: This module makes available real storage manager resources associated with a terminating address space. It also contains the SRB purge exit for all RSM SRBs.

Entry from:

For entry point IEAVDLAS: IEAVTERM.

For entry point IEAVSRBP: IEAVEPD0 or IEAVTMMT.

Exit to: Caller.

IEAVDSEG — Destroy Segment

Operation: This module destroys one segment table entry by invalidating it.

Entry from: IEAVRELS and IEAVEQR.

Exit to: Caller.

IEAVEAC0 — ASCB Dispatching Queue Manipulation Routine

Operation: This routine moves an ASCB to a new dispatching priority, adds an ASCB to a specified priority, or deletes a specified ASCB from the dispatching queue.

Entry from:

For entry point IEAVEAC0: Memory create/delete, SRM, and swap out/in.

For entry point IEAVEAC3: RTM, for functional recovery routine processing.

Exit to: Caller.

IEAVEAT0 — ATTACH Service Routine

Operation: This module creates a new task as a subtask of an existing task. ATTACH can also set an indicator in a TCB for LINK, recover the TCB dispatching queue for RTM, and save error information for RTM when the ATTACH FRR has not yet been entered.

Entry from:

For entry point IGC0004B: SVC interrupt handler to create a TCB, an XSB, and an SVRB.

For entry point IGC043R1: The dispatcher to obtain and initialize save area that LINK uses for the TCB being attached.

For entry point IGC042R2: RTM to ensure that the FRR is in the correct address space and has not been used.

For entry point IGC042ES: RTM to save error information in the SDWAVRA.

Exit to:

For entry point IGC0004B: Exit prolog to return control to the caller before reentering ATTACH at IGC042R1.

For entry point IGC042R1: LINK (IEAQCS01).

For entry point IGC042R2: RTM.

For entry point IGC042ES: RTM.

IEAVEBBR — Bind Break Service Routine

Operation: This module is called in two situations:

- When a module changes the cross memory environment (the AX, LX, or LT) of an address space, it calls IEAVEBBR to determine if the change made the cross memory environment of any online processor invalid. If it did, the unit of work on that processor is abended. If a processor's cross memory environment is still valid, IEAVEBBR ensures that the processor's cross memory control registers contain current information.
- When an address space can no longer be accessed in cross memory mode, the module handling the situation calls IEAVEBBR to mark the address space invalid for cross memory access and to break all active binds to the address space. If any online processor's primary or secondary address space is the same as the address space that is no longer accessible, the unit of work on that processor is abended.

Entry from:

For entry point IEAVEBBR: A module that changes the cross memory environment of an address space.

For entry point IEAVEBB2: IEAVEES.

Exit to: Caller.

Called routines: IEEVEXSN and IEAVECMS at entry point IEAVLACB.

IEAVECH0 — CHAP Service Routine

Operation: This module alters the dispatching priority of a task.

Entry from: SVCIH, at entry point IGC044.

Exit to: EXIT prolog (IEAVEEXP) to force dispatcher-entry.

IEAVECMS — CMSET Service Routine

Operation: This module performs the following services to satisfy the macro indicated:

- Establishes a specified address space as both the caller's primary and secondary address space (CMSET SET)
- Restores a previously existing cross memory environment (CMSET RESET)
- Establishes a specified address space as the secondary address space, and puts the caller into secondary addressing mode (CMSET SSARTO)
- Restores the secondary address space and cross memory addressing mode that existed before an associated CMSET SSARTO macro executed (CMSET SSARBACK)
- Locates the returns the address of an identified ASCB (LOCASCB)
- Provides branch entry to the dispatcher (CALLDISP)

Entry from:

For entry point IEAVCMS1: Issuer of a CMSET SET macro.

For entry point IEAVCMR1: Issuer of a CMSET RESET, CHKAUTH=YES macro.

For entry point IEAVCMR2: Issuer of a CMSET RESET, CHKAUTH=NO macro.

For entry point IEAVCMST: Issuer of a CMSET SSARTO macro.

For entry point IEAVCMSB: Issuer of a CMSET SSARBACK macro.

For entry point IEAVLACB: Issuer of a LOCASCB macro.

For entry point IEAVCDEN: Enabled issuer of a CALLDISP macro.

For entry point IEAVCDDS: Disabled issuer of a CALLDISP macro.

Exit to:

For all entry points except IEAVCDEN and IEAVCDDS: Caller by issuing a BR 14 instruction.

For entry points IEAVCDEN and IEAVCDDS: Dispatcher at entry point IEAV0DS1.

Error exit: RTM via an ABEND macro.

IEAVEDR — Interprocessor Communication Direct Signal Routine

Operation: This module gives the interface needed to signal other processors that invoke any one of the twelve functions of: sense, external call, emergency signal, start, stop, restart, initial program reset, program reset, stop and store status, initial microprogram load, Initial CPU reset, and CPU reset.

Entry from: IEAVEDR via BALR.

Exit to: IEAVEDR for normal exit; and to a System Termination Facility for abnormal exit.

IEAVEDSR — Dispatcher Recovery

Operation: This module calls the necessary functions to clean up dispatching queues and the processor in order to continue to dispatch ready work.

Entry from: IEAVESPR.

Exit to: IEAVERTN, return address in IEAVESPR (super FRR).

IEAVEDS0 — Dispatcher

Operation: This module dispatches tasks, local supervisor routines and SRBs.

Entry from:

For entry point IEA0DS1 (general entry point): SVC FLIH, EXIT prolog, ACR, STATUS (status, stop, synch), and CALLDISP (type 6) when cross memory status has already been saved.

For entry point IEAODS (general entry point): CALLDISP (type 6) when cross memory status has not been saved.

For entry point IEAVDSPC: Program FLIH and lock manager (IEAVELK).

For entry point IEAVDSTC: IEAVETCL to dispatch a task.

For entry point DDSRBRTN: SVC FLIH when a type 6 SVC exits and requests that control be given to an SRB.

For entry point IEAPDS7: I/O and external FLIH when an unlocked task is being preempted.

For entry point IEAPDS7A: I/O and external FLIH when a task that holds a lock is being preempted.

For entry point IEAPDS7B: I/O and external FLIH when no status needs to be saved.

For entry point IEAPDS7C: I/O and external FLIH when a task that holds a CML lock is being preempted.

For entry point IEAPDSPT: All SRB routines.

For entry point DSJSTCSR: SVC FLIH (IEAVESVC), timer SLIH (IEAVRTIO), EXIT (IGC003), and SRB time limit initialization routine (IEAVRT03).

Exit to: Global SRB/SSRB dispatcher, local SRB/SSRB dispatcher, local supervisor dispatcher, task dispatcher, or wait task dispatcher.

Called routines: IEAVESC5, IEAOEF03, IEAVRSPN, IEAVEMS0, TRWT, TRSRB2, IEADISP1, IEADISP2, IEADISP3, IEAVRJLM.

IEAVEED0 — DETACH service routine

Operation: This module detaches a task and frees its resources such as the TCB and any associated program save areas.

Entry from:

For entry point IGC062: SVC IH.

For entry point IGC062R1: Supervisor routine via branch entry.

Exit to:

For entry point IGC062: Caller via exit prolog.

For entry point IGC062R1: Caller.

IEAVEEEP — IQE Purge Routine

Operation: Removes the IQEs for the terminating task from the asynchronous exit queue.

Entry from: End-of-task processing or ABEND processing.

Exit to: Caller.

IEAVEEER — Stage 3 Exit Effector Recovery Routine

Operation: This module uses the queue verifier IEAVEQV0 to verify and correct the asynchronous exit queues.

Entry from: Dispatcher recovery (IEAVEDSR), at entry point IEAVEEER.

Exit: Caller via a BR14.

IEAVEEE0 — Stage 3 Exit Effector

Operation: This module dequeues IQEs, RQEs, and SRBs from the asynchronous exit queues, and it also schedules the corresponding asynchronous exit by queueing the associated IRB to the TCB.

Entry from: The dispatcher at entry point IEA0EF03.

Exit to: Caller via register 14.

IEAVEEE2 — Asynchronous Exit Queue Handler (Stage 2)

Operation: This module places on the correct queue a request to schedule an asynchronous exit.

Entry from: A system routine that passes an IQE, SRB, or an RQE to IEAVEEE2.

Exit to: Caller.

IEAVEES — Emergency Signal Second Level Interrupt Handler

Operation: This module receives control from the external FLIH and gives control to the appropriate receiving routine whenever emergency signal interrupt occurs.

Entry from: IEAVEEXT by using a BALR 2, 10.

Exit to: Caller.

IEAVEEXP — EXIT Prolog Routine

Operation: This routine performs exit functions for all SVC routines.

- At entry point IEAVEXPR it provides normal SVC exit functions for all SVC routines unless they are the last RB on the TCB.
- At entry point IEAVEXP1 it provides SVC routines the ability to enter the dispatcher after exit processing.
- At entry point IEAVEXP2, it provides SVC exit functions for microcode assisted type-1 SVC routines using the cross memory facility.
- At entry point EXPEPAT6, it provides SVC exit functions for type-6 SVC routines which specify RETURN=CALLER.

Entry from:

For entry point IEAVEXPR: All SVC routines.

For entry point IEAVEXP1: All SVC routines.

For entry point IEAVEXP2: Microcode assisted type-1 SVC routines using the cross memory facility.

For entry point EXPEPAT6: The SVC FLIH, IEAVESVC.

Exit to:

For entry point IEAVEXPR: Caller if this is not the last RB or to IEAVEOR if this is the last RB.

For entry point IEAVEXP1: Dispatcher.

For entry point IEAVEXP2: Caller or dispatcher.

For entry point EXPEPAT6: Caller or dispatcher.

Called routines: IEAVESS, IEAVTSBP.

IEAVEEXT — External First Level Interrupt Handler (FLIH)

Operation: This interrupt handler routine saves the status of the interrupted program, and decides what kind of external interrupt occurred in order to give the

proper SLIH control. It decides whether to return to the interrupted program or to the dispatcher.

Entry from: An external new PSW as a result of PSW switch caused by an external interrupt.

Exit to: The interrupted task, SRB, or the dispatcher.

IEAVEE1R — External FLIH Recovery Routine 1

Operation: This module cleans up the external FLIH resources to allow ABEND to percolate the error.

Entry from: Super FRR (IEAVESPR) if an error occurs during operation of the external FLIH.

Exit to: RTM via an ABEND SVC.

IEAVEE2R — External FLIH Recovery Routine 2

Operation: This module cleans up the FLIH resources to allow ABEND to percolate the error.

Entry from: Super FRR (IEAVESPR) if an error occurs during operation of a recursive entry into the external FLIH.

Exit to: RTM via an ABEND SVC.

IEAVEE3R — External FLIH Recovery Routine 3

Operation: This module cleans up the external FLIH resources to allow ABEND to percolate the error.

Entry from: Super FRR (IEAVESPR) if an error occurred during the processing of a 2nd level of recursion into the external FLIH.

Exit to: RTM via an ABEND SVC.

IEAVEF00 — CIRB (Create IRB)

Operation: This module acquires space for an IRB, formats it according to the input parameters, and returns its address to the caller. It might also acquire space for a work area, which is appended to the end of the IRB.

Entry from: SVC interrupt handler (IEAVESVC).

Exit to: If branch entered, the caller by issuing a BR14; if entered by ICAVESVC, EXIT prolog (IEAVEEXP).

IEAVEINT — Intersect Service Routine

Operation: This module is called when a set request is made to obtain the local or global intersect, ensuring serialization with the dispatcher or when a reset request is made to release dispatcher serialization requirements.

Entry from:

For entry point IEAVEINT: INSECT macro via a BALR (for a local SET request) or via a BAL (for a global SET request).

For entry point IEAVEINR: INSECT macro via a BALR (for a local intersect RESET request).

For entry point IEAVINTR: IEAVTRTS via an LPSW instruction.

Exit to: Caller.

Called routines: IEAVEXSN.

IEAVEIO — I/O First Level Interruption Handler

Operation: This module saves the status of the interrupted program, invokes IOS to process the interruption, and determines whether to return to the interrupted program or to the dispatcher.

Entry from: I/O new PSW as a result of a PSW switch caused by an I/O interrupt.

Exit to: Interrupted task, SRB, or dispatcher.

IEAVEIOR — I/O FLIH Recovery Routine

Operation: This module makes the I/O FLIH recovery operable by clearing the I/O FLIH recursion indicator, allowing ABEND to percolate to one previous level of recovery.

Entry from: Super FRR (IEAVESPR).

Exit to: IEAVESVR at entry point IEAVEABD (common ABEND) via register 15.

IEAVEIPR — IPC Functional Recovery Routine

Operation: This module clears RISGNL buffers and indicators so that RISGNL can be invoked again.

Entry from: IEAVTRTS.

Exit to: RTM via BR14.

IEAVELCR — Low Core Refresh

Operation: This module examines and correctly sets constants and addresses in selected system control blocks for continual system execution.

Entry from: IEAVTRT1.

Exit: Via BR14 to the lock manager verification (IEAVELKR).

IEAVELK — SETLOCK Service Routine

Operation: This module sets and releases the various system locks by manipulating the lockwords associated with them. Locking synchronizes and controls access to the serially reusable resources of the system.

Entry from: A system routine that issued a SETLOCK macro instruction.

Exit to: Caller.

IEAVELKR — Lock Manager Verification

Operation: This module corrects correlations between the locking hierarchy mask and the lock words. When a restart condition and a lock spin condition exists, it clears spin lockwords.

Entry from:

For entry point IEAVELKR: IEAVELCR to verify and reconstruct locks, the lock table, and the hierarchy mask.

For entry point IEAVLKRR: Recovery routine for lock manager to determine if SETLOCK recovery is applicable, and if so, whether a retry or percolation situation exists.

Exit to:

For entry point IEAVELKR: RSM recovery routine (IEAVRELS).

For entry point IEAVLKRR: RTM or ASVT reconstruction routine (IEAVVFRR).

IEAVEMCR — Memory Create

Operation: This module establishes memory addressability, enqueues the new ASCB, and schedules a service request on the service priority list.

Entry from: IEDAY3 via XCTL, IEEVWAIT, IEEMB881.

Exit to: Caller.

IEAVEMDL — Memory Delete

Operation: This module dequeues the ASCB of the specified address space from the ASCB ready queue, informs SRM of the address space deletion, determines if the ASID can be reused, adds this ASCB to a queue of ASCBs waiting to be freed, and releases storage for all eligible ASCBs.

Entry from: IEAVTMMT.

Exit to: IEAVTMMT.

Entry point MDLESTAE: This entry point frees any ASCBs on the queue of ASCBs waiting to be freed and page frees the storage for IEAVEMDL, if it is fixed.

Entry from: RTM.

Exit to: RTM.

IEAVEMIN — Memory Initialization

Operation: This module creates a dispatchable TCB/SVRB on an ASCB's TCB ready queue. It also sets the SVRB to enter the program manager for linkage to the RCT.

Entry from: IEAVEMCR.

Exit to: IEAVAR00.

IEAVEMRQ — Memory Request

Operation: This module assigns an ASID for a new address space. It creates the ASCB and notifies the system resource manager that a new address space is being created.

Entry from: IEE0803D, IEEMB881.

Exit to: IEE0803D.

IEAVEMS0 — Memory Switch Routine

Operation: This module notifies the dispatcher that it is to either switch to another address space or begin searching for work on the ASCB dispatching queue.

Entry from: Supervisor routines that recognize more work has been made ready. These include IEAVELK, IEAVECHO, IEAVESC0, or IEAVSY50.

Exit to: Caller.

IEAVENQ1 — Queue Manipulation Routine See ISGGNQDQ. (IEAVENQ1 is an alias entry point for ISGGNQDQ.)

IEAVEOR — Exit

Operation: This module terminates the current RB or SPIE exit. In RB termination it also recognizes end-of-task and thereupon initiates and later terminates EOT processing.

Entry from: User and system programs.

Exit to: EXIT prolog via entry register 14 or by normal task termination via SVC 13.

Called routines: IEAVSETS, IEAVESSE, IEAVEBBR, IEAVTSBP, IEAPPGMX, IEAVESS, IEAQSPET, IGC062R1.

IEAVEPCO — DAT–off Section of the Program First Level Interrupt Handler (FLIH)

Operation: After a program check occurs, the program FLIH receives control from the program check new PSW. It determines the program check type and routes control to the appropriate second level interrupt handler to process it. The program FLIH's code is located in two modules: IEAVEPO, which executes with DAT disabled and without virtual addressing, and IEAVEPC, which executes with DAT enabled. While performing the program FLIH functions, the two modules pass control back and forth using an LPSW instruction.

Entry from: Program check new PSW at entry point IEAQPK00.

Exit to:

- Interrupted program by issuing an LPSW instruction. This exit is used when the interrupt was:
  - A PER or space switch interrupt and IEAVTPER did not request that the program be terminated
  - A page fault and the paging supervisor was able to satisfy the page fault without I/O
  - A recursive PER or space switch interrupt.
- The dispatcher at entry point IEA0DS when the program FLIH scheduled an SRB to perform PIE/PICA processing
- The dispatcher at entry point IEAVDSPC when a page exception has occurred and the paging supervisor has indicated that I/O is to be done.
- IEAVTRT1 at entry point PROGCK in these situations:
  - A PER or space switch interrupt occurred and IEAVTPER requested that the program be terminated
  - A program check interrupt occurred and there was no active SPIE environment
  - An addressing exception, invalid page fault, unexpected error in RSM, or a translation specification exception occurred.
- Super SPIE routine when a page exception has occurred and the interrupted program has an active super SPIE

Error exit: IEAVTRT1 at entry point PROGCK, IEAVTRT1 at entry point DATERR, or IGFPTERM.

Called routines: IEAVEXMS.

IEAVEPC — DAT–on Section of the Program First Level Interrupt Handler (FLIH)

Operation: The program FLIH's operation is described in the module description for IEAVEPCO.

Entry from: The DAT-off section of the program FLIH at entry point IEAVPCON.

Exit to: See the module description for IEAVEPCO.

Error exit: See the module description for IEAVEPCO.

Called routines: AHLMCIH, IEAVDSPC, IEA0DS, IEAVPSRB, IEAVSPER, IEAVPIX2, IEAVTPER, IEAVTRT1, IEAVESC0, and IEAVECMS.

IEAVEPCR — Program FLIH Recovery Routine

Operation: This module clears recursion indicators and allows percolation of recovery.

Entry from: Super FRR (IEAVESRR) for recovery of program check FLIH, program interrupt handler (IEAVEAC), or IEAVEPCO.

Exit to: RTM via on ABEND SVC.

IEAVEPDR — PURGEDQ Recovery Routine

Operation: This module performs FRR and ESTAE functions for PURGEDQ and itself.

Entry from:

For entry point IEAVEPDF: FRR for PURGEDQ, which handles errors during queue manipulation or scanning in PURGEDQ.

For entry point IEAVEPDE: ESTAE for PURGEDQ

For entry point IEAVEPDS: Retry routine for PURGEDQ.

Exit to:

For entry point IEAVEPDE: RTM via BALR.

For entry point IEAVEPDF: RTM via BALR.

For entry point IEAVEPDS: PURGEDQ at main entry point IGC123.

IEAVEPD0 — PURGEDQ Service Routine

Operation: This routine removes specified SRBs from the service manager queues. It ensures that the suspended SRBs have completed their processing before dequeueing them.

Entry from: A supervisor routine.

Exit to: Caller.

IEAVEQR — V=R Allocation

Operation: This module allocates and frees contiguous real storage for use in V=R regions.

Entry from:

For entry points IEAVEQR and IEAVEQRF: IEAVPRTO.

For entry point IEAVEQRI: IEAVPFTE.

For entry point IEAVEQRC: IEAVEQRI and IEAVSQA.

For entry point IEAVEQRP: For SRB page-out.

Exit to: Caller.

Error exit:

From entry point IEAVEQRC: IEAVRCV.

From entry point IEAVEQRP: IEAVRCV or IEAVRMT.

IEAVEQV0 — Queue Verifier

Operation: The queue verifier (QV) verifies and corrects queue structures. It performs three basic functions:

- If the queue structure has been destroyed, QV reconstructs the queue from any available information.
- Once the queue structure has been verified or reconstructed, QV will remove any elements that contain bad data.
- Records all detected errors, and corrective actions taken in an output data area.

Entry from:

For entry point IEAVEQV1: Supervisor recovery routines for single-threaded queue with header only (type 1).

For entry point IEAVEQV2: Supervisor recovery routines for single-threaded queue with header and trailer (type 2).

For entry point IEAVEQV3: Supervisor recovery routines for double-threaded queue (type 3).

Exit to: Caller.

IEAVERER — Restart IH Recovery Routine

Operation: This module makes the restart FLIH recovery function operable.

Entry from: Super FRR (IEAVESPR) at entry point IEAVERER.

Exit to: IEAVESVR at entry point IEAVEABD to issue an abend.

IEAVERES — Restart Interrupt Handler

Operation: This entry saves the status of the interrupted program and proceeds to restart FLIH extension (IEAVEREX).

Entry from: The restart new PSW.

Exit to: RTM via IEAVEREX or resume interrupted program.

Called routines: IEAVEREX.

IEAVEREX — Restart FLIH Extension

Operation: This module processes the restart interrupt fielded by IEAVERES. Processing includes handling the operator-selected restart reason on processor consoles that support the restart reason function. Depending on the selected reason, this module:

- Abends the interrupted program after displaying job information to the operator for verification.
- Performs high-level system diagnosis and repair in the following areas:

  system nondispatchability
  exhausted WTO buffer elements
  no active batch or TSO jobs
  I/O problems (via IEACVRSTS)

Entry from: IEAVERES.

Exit to: IEAVERES to resume interrupted work, RTM if interrupted work is to be abended.

Called routines: IEEVDCCR, IEAVSETS.

Entry point: IEAVERXI (this entry point is a non-executable text string to be displayed on processor consoles that support the restart text function. A pointer to this entry point is contained in the CVT (CVTRESTX). IPL/NIP and VARY CPU, ONLINE interface with the console and pass this text.)

IEAVERI — Interprocessor Communication Remote Immediate Service Routine

Operation: This module gives the interface needed to signal other processors. It is used to invoke a specific program on any of the online MP-configured processors. It generates an emergency signal to the specified processor which then routes control to the specified program.

Entry from: RISGNL macro via BALR.

Exit to: Caller for normal exit.

IEAVERP — Interprocessor Communication Remote Pendable Service Routine

Operation: This module gives the necessary interface to signal other processors. It is used to invoke a specific program on any of the online

MP-configured processors; it generates an external call to the specified processor, which will then routes control to the specific program.

Entry from: RPSGNL macro via BALR.

Exit to: Caller for normal exit.

IEAVESCR — SCHEDULE Recovery Routine

Operation: This module:

- Verifies the SRB journal queue and reschedules any remaining SRBs.
- Uses the queue verifier (IEAVEQV0) to verify the GSPL and LSPL queues for every address space in the system.

Entry from:

For entry point IEAVESCR: Dispatcher recovery (IEAVEDSR) to clean up for the module (IEAVESC0).

For entry point IEAVESQV: PURGEDQ FRR (IEAVEPDR) for queue verification.

Exit to:

For entry point IEAVESCR: Dispatcher recovery.

For entry point IEAVESQV: PURGEDQ FRR.

IEAVESC0 — SCHEDULE Service Routine

Operation: For local SRBs, the SRB is placed on the appropriate queue, a waiting processor is signalled, and the system resource manager is invoked when work has become ready for an address space that is swapped out. For global SRBs, it is entered only if there is a waiting processor.

Entry from:

For entry point IEAVESC1: Issuer of a local enabled SCHEDULE macro.

For entry point IEAVESC2: Issuer of a local disabled SCHEDULE macro or IEAVESC1.

For entry point IEAVESC3: Issuer of an enabled global SCHEDULE macro.

For entry point IEAVESC4: Issuer of a disabled global SCHEDULE macro or IEAVESC3.

For entry point IEAVESC5: Dispatcher (IEAVEDS0).

For entry point IEAVESC6: Lock manager (IEAVELK).

Exit to:

For entry point IEAVESC1: IEAVESC2.

For entry points IEAVESC2, IEAVESC4, IEAVESC5, and IEAVESC6: Caller.

For entry point IEAVESC3: IEAVESC4.

Error exit: RTM via an ABEND macro.

Called routines: IEAVEMS0.

IEAVESPI — SPIE SRB Routine

Operation: After a program check has occurred for an unlocked task with a SPIE routine, IEAVESPI receives control to initialize a TCB/RB and a PIE to enter the SPIE routine. IEAVESPI marks the task dispatchable and IEAVEDS0 later dispatches it.

Entry from: An SRB scheduled by IEAVEPC.

Exit to: Dispatcher (IEAVEDS0).

IEAVESPM — SRB/SSRB Pool Manager

Operation: IEAVESPM processes four macros, GETSRB, GETSSRB, FREESRB, and FREESSRB. Depending on which macro is issued. IEAVESPM:

- Obtains an SRB and a six-word parameter area from the SRB pool in subpool 245. If the pool is empty and the caller specified the UNCOND or EXPAND option, IEAVESPM either expands the pool or uses GETMAIN to obtain a single SRB and parameter area (GETSRB). IEAVESPM also initializes the SRB.
- Obtains and initializes an SSRB and XSB from the SSRB pool in subpool 239. If the pool is empty and the caller specified the UNCOND or EXPAND option, IEAVESPM either expands the pool or uses GETMAIN to obtain a single SSRB/XSB (GETSSRB).
- Frees an SRB and its associated parameter area. If IEAVESPM obtained the storage from the SRB pool, it returns the storage to the pool. Otherwise, IEAVESPM used GETMAIN when obtaining the storage, so it frees the storage (FREESRB).
- Either returns an SSRB/XSB to the SSRB pool, or frees the storage (FREESSRB). (See the above explanation.)

Entry from:

For entry point IEAVSPM1: A GETSRB macro via a BALR 14, 15 instruction.

For entry point IEAVSPM2: A GETSSRB macro via a BALR 14, 15 instruction.

For entry point IEAVSPM3: A FREESRB macro via a BALR 14, 15 instruction.

For entry point IEAVSPM4: A FREESSRB macro via a BALR 14,1 5 instruction.

For entry point IEAVEFRE: PURGEDQ to free the storage allocated for an SSRB/XSB.

For entry point IEAVSPMR (the FRR routine): RTM to repair the SRB and SSRB pools and release the SALLOC lock (if held).

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: Lock manager (IEAVELK), GETMAIN/FREEMAIN (IEAVGM00), and queue verifier (IEAVEQV0).

IEAVESPR — Supervisor Control FRR (Super FRR)

Operation: This module is a router routine for supervisor control. It determines retry and recovery addresses. In particular it can cause control to be routed to:

Function	Module name
Restart FLIH recovery	IEAVERER
Program FLIH recovery	IEAVEPCR
External FLIH 1 recovery	IEAVEE1R
External FLIH 2 recovery	IEAVEE2R
External FLIH 3 recovery	IEAVEE3R
I/O FLIH recovery	IEAVEIOR
SVC FLIH recovery	IEAVESVR
Dispatcher mainline	IEA0DS (default)

Entry from:

For entry point IEAVESPR: RTM for recovery control by means of an LPSW, if an error occurred while one of the supervisor control FRR stacks was in control.

For entry point IEAVERTN: Mainline IEAVESPR.

Exit to: RTM, which will route control to one of the functions mentioned above.

IEAVESRT — STOP/RESET Service Routine

Operation: This module saves the status and performs the functions necessary to suspend a task or an SRB, and restores the status and performs the functions necessary to make a suspended task or SRB redispatchable.

Entry from:

For entry point IEAVSUSP: Paging supervisor to stop a task or SRB.

For entry point IEAVSUSQ: From a disabled routine to stop a task or SRB.

For entry point IEAVRSTD: From a disabled routine to make a stopped task or SRB redispatchable.

For entry point IEAVRSET: From the paging supervisor to make a stopped task or SRB redispatchable.

For entry point STOPFRR: RTM (IEAVTRT1).

For entry point RESETFRR: RTM (IEAVTRT1).

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEAVEMS0.

IEAVESSE — Space Switch Event Mask Manager

Operation: This module enables and disables the space switch event mask (ASTESSEM) contained in the address space second table (ASTE). When the mask is enabled, a space switch program interrupt occurs whenever a program call (PC) or program transfer (PT) instruction switches addressability to or from an address space. There is one mask per address space, and several users per mask. Each user has a bit in the space switch ownership word (ASCBSSOM), which, when on, indicates that the associated user requires that the mask be enabled. IEAVESSE's caller indicates which bit is to be set or cleared. After IEAVESSE updates the ownership word, it checks whether any users still require an enabled mask. IEAVESSE then either disables the EVENT mask by setting it to zero, or enables it by setting it to one.

Entry from:

For entry point IEAVESSE: IEAVTSSH, IEAVTLCL and IEAVEOR.

Exit to: Caller.

Error exit: RTM via an ABEND macro (completion code X'058') when the input ASID is invalid, the input index into the ownership word is invalid, or the reserved bits in the input registers are not zero.

Called routines: IEAVECMS at entry point IEAVLACB (LOCASCB).

IEAVESSI — Service Routine that Sets Data in, or Obtains Data from an Entry in the Subsystem Affinity Table

Operation: This module sets a data word or obtains the data word from a specified entry in a TCB's subsystem affinity table (SSAT).

Entry from:

For entry point IEAVESAS: Issuer of a SSAFF SET macro.

For entry point IEAVESAF: Issuer of a SSAFF OBTAIN macro.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: TCB treescan routine in IEAVSETS (IEATRSCN), IEAVELK.

IEAVESTS — SRB Status SAVE/RESTORE/MODIFY Routine

Operation: IEAVESTS processes SRBSTAT macros. Depending on whether the macro specifies SAVE, RESTORE, or MODIFY, IEAVESTS:

- Saves the current status or an SRB in a specified SRB status save area (STSV). The saved status includes:
  - General purpose and floating point registers
  - Processor and SRB timing information
  - Processor affinity
  - The purge ASID and the purge TCB address
  - The PCLINK stack header
  - Cross memory status
  - The normal FRR stack

  If the caller provided the address of an FRR routine, IEAVESTS also adds the FRR to the normal FRR stack. (SRBSTAT SAVE)

- Restores status from values saved in a specified STSV. (SRBSTAT RESTORE)

- Stores a new purge ASID and purge TCB address in either the STSV or the LCCA. (SRBSTAT MODIFY)

Entry from:

For entry point IEAVESTS: An SRBSTAT SAVE macro via a BALR 14, 15 instruction.

For entry point IEAVESTR: An SRBSTAT RESTORE macro via a BALR 14, 15 instruction.

For entry point IEAVESTM: An SRBSTAT MODIFY macro via a BALR 14,1 5 instruction.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

IEAVESVC — SVC First Level Interrupt Handler

Operation: The SVC interrupt handler decides whether the issuer can issue SVCs, if the issuer is authorized, and what type of SVC requested. It also sets up the proper input environment for the SVC routine.

Entry from: SVC new PSW as a result of a PSW switch caused by issuing an SVC instruction.

Exit to: RTM (SVCERR), SVC routine.

Error exit: SVC 13, IEAVEXPR (EXIT prolog).

IEAVESVR — SVC IH Recovery Routine

Operation: This module performs two major functions:

- It makes the SVC FLIH recovery operable by clearing the recursion indicator.

- It issues ABENDs for SVC, I/O FLIHs.

Entry from:

For entry point IEAVESVR: Super FRR (IEAVESPR) if an error occurred during the operation of SVC FLIH.

For entry point IEAVEABD: SVC FLIH recovery routine, I/O FLIH recovery routine (IEAVIOR).

Exit to:

For entry point IEAVESVR: Falls through to entry point IEAVEABD.

For entry point IEAVEABD: ABEND.

IEAVETCL — SUSPEND/RESUME/TCTL Processor

Operation: This module contains the routines that suspend a TCB/RB, resume a TCB/RB, and transfer control to a TCB/RB.

Entry from:

For entry point IEAVSPND: Issuer of a SUSPEND macro.

For entry point IEAVRSUH: Issuer of an unconditional synchronous RESUME macro.

For entry point IEAVPSUS: Issuer of an unconditional synchronous RESUME macro in a specified address space.

For entry point IEAVPSUA: Issuer of an unconditional asynchronous RESUME macro.

For entry point IEAVRSCS: Issuer of a conditional synchronous RESUME macro.

For entry point IEAVRSCA: Issuer of a conditional asynchronous RESUME macro.

For entry point IEAVTCTL: Issuer of a TCTL macro.

For entry point IEAVFTCR: RTM.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEAVEMS0 and IEAVEDS0.

IEAVEVAL — Validity Check

Operation: This module validates an address or address range by determining if the storage key and the protect key match.

Entry from: Supervisor routines via branch.

Exit to: Caller.

IEAVEVRR — ASVT and AFT Verification and Reconstruction Routine

Operation: This module rebuilds the real address space first table (AFT) from the virtual AFT and restores control register 14 (the pointer to the real AFT). It then checks for invalid entries in the ASVT and, if any exist, rebuilds the ASVT.

Entry from:

For entry point IEAVEVRR: RSM recovery routine (IEAVRELS).

For entry point IEAVVFRR: Lock repair routine's FRR (IEAVLKRR).

Exit to: RTM.

IEAVEVT0 — EVENTS Service Routine

Operation: This module performs three services:

- Event table creation and deletion, which causes an event table to be queued or dequeued from the requesting TCB.
- ECB initialization, which causes the ECB to be initialized to the event format. If the ECB is already posted, IEAVEVT0 adds the ECB address to the event list of completed ECBs.
- Event table wait, which causes the caller to stop processing and wait for the completion of one of n events.

Entry from:

For entry point IEAVEVT0: Supervisor using branch entry to perform WAIT and/or ECB initialization.

For entry point IGC125: SVC interrupt handler to perform WAIT and/or ECB initialization.

For entry point IEAVEVT1: SVC interrupt handler through the type 2 extended SVC router to perform event table creation or deletion.

Exit to:

For entry point IEAVEVT0: Dispatcher (IEA0DS) or caller.

For entry point IGC125: EXIT prolog or ABEND.

For entry point IEAVEVT1: EXIT prolog or ABEND.

IEAVEXMS — Cross Memory Second Level Interrupt Handler

Operation: When an X'0C1' or an X'0C2' program interrupt occurs, the program FLIH (IEAVEPC) branches to this module. If a processor without cross memory hardware installed caused the interrupt by issuing a cross memory instruction, IEAVEXMS routes control to the appropriate simulation subroutine to simulate the instruction. Otherwise, it returns to IEAVEPC, which handles the interrupt.

Entry from:

For entry point IEAVEXM1: IEAVEPCO.

For entry point IEAVEXM2: IEAVEPCO when a recursive program interrupt occurs.

Exit to: Interrupted program.

Error exit: IEAVEPCO.

Called routines: IEAVEPC, IEAVELK, and IEEVEXSN.

IEAVEXS — External Call Second Level Interrupt Handler

Operation: Upon an external call interruption, this module receives control from the external FLIH. It routes control to the appropriate receiving routine.

Entry from: External FLIH via BALR 2, 10.

Exit to: Caller.

IEAVFP — Find Page

Operation: This module returns the virtual addresses of the page and external page table entries corresponding to the input virtual address provided.

Entry from:

For entry point IEAVFP1: Non-RSM routines.

For entry point IEAVFP2: Other RSM routines.

Exit to: Caller.

IEAVFRCL — FREECELL Routine

Operation: This module returns a cell to the quickcell pool.

Entry from: A routine that has no further need for allocated quickcell.

Exit to: Caller.

IEAVFREE — PGFREE

Operation: This module executes a PGFREE request or backs out a partially completed PGFIX request that is being cancelled.

Entry from:

For entry point IEAVFREE: IEAVFXLD, IEAVPSI.

For entry point IEAVANYW: IEAVPSI.

Exit to: Caller.

IEAVFRLK — Lock Freeing Routine

Operation: Depending on whether the processor holds the dispatcher or the local lock, this module clears either the global or local intersect in the SVT. It then releases all locks held by the caller. If the current task is the target of an earlier status request that could not be completed until a CML lock was released, IEAVFRLK calls the STATUS routine (entry point IGC07904 in IEAVSETS) to complete the pending status request.

Entry from: A caller in supervisor state, zero protect key.

Exit to: Caller.

IEAVLKRM — Lock Resource Manager

Operation: When an address space is being terminated, this module receives control either to:

- Release any CML locks owned by the terminating address space (entry point IEAVLKRM)
- Reschedule any SRBs or SSRBs on the local lock SRB suspend queue. This puts the SRB/SSRBs back on the local or global SPLs so that the PURGEDQ routine finds them and calls their respective resource termination managers (entry point IEAVELRM).

Entry from:

For entry point IEAVLKRM: RTM memory termination processing.

For entry point IEAVELRM: RTM.

For entry point LKRMFRR: RTM.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEAVESC0.

IEAVFXLD — Page Fix/Page Load

Operation: This module brings virtual storage pages into real storage if they are not already there. For PGFIX requests, it fixes the pages in real storage so that no paging I/O will be performed for them.

Entry point: IEAVFXLD from IEAVPSI.

Exit to: IEAVPSI.

Entry point: IEAVFXL from any RSM routine that decreases the PCB count in a PGFIX or PGLOAD root PCB to zero.

Exit to: Caller.

IEAVGCAS — Create/Free Address Space Routine, VSM Task Termination Routine

Operation: This routine establishes addressability for a new address space, frees task-related storage at task termination, and frees VSM global storage at address space termination.

Entry from:

For entry point IEAVGCAS: IEAVEMCR.

For entry point IEAQSPET: IEAVEQR.

For entry point IEAVGFAS: IEAVTSKT.

Exit to: Caller.

IEAVGFA — General Frame Allocation

Operation: This module performs real frame allocation services, including reclamation. Paging I/O is started where necessary to retrieve an auxiliary storage copy of a page.

Entry from:

For entry point IEAVGFA: IEAVPIX, IEAVFXLD, IEAVSWIN, or IEAVGFD2.

IEAVGFAX — General Frame Allocation Extension

Operation: This module contains many service routines called by IEAVGFA and IEAVSQA (as well as other routines in IEAVGFAX) for frame allocation.

Entry from:

For entry point IEAVAIAL: IEAVGFA and IEAVS1BK to dispose of PCBs on an ASM I/O request area (AIA) chain.

For entry point IEAVCPCB: IEAVGFA to copy a PCB to another PCB.

For entry point IEAVEQRS: IEAVGFA and IEAVSQA to find the best V=R frame on the AFQ that is below the 16 megabyte (Mb) line.

For entry point IEAVFIXC: IEAVGFA and IEAVGFRAM to update the PFTE fix count and the system counts of fixed pages.

For entry point IEAVFRAM: IEAVGFA to associate a page and a frame.

For entry point IEAVFXBK: IEAVGFA and IEAVRELA to end a fix request that could not be satisfied.

For entry point IEAVGFD2: IEABPFTE via IEAVEDS0 to drive allocation for deferred requests.

For entry point IEAVPCPY: IEAVGFA and IEAVSQA to copy a page from one frame to another.

For entry point IEAVPDIS: IEAVGFA to dispose of a PCB for an immediate allocation.

For entry point IEAVPEX: IEAVGFA to exchange the contents of two frames.

For entry point IEAVPGIN: IEAVGFA to queue a PCB awaiting start of a page-in.

For entry point IEAVPUTD: IEAVGFA, IEAVAIAL, IEAVFXBK and IEAVS1BK to put a PCB on the GFA defer queue.

For entry point IEAVRELA: IEABGFA to relate a PCB to queued requests.

For entry point IEAVRSMG: ERBMFETR to count the number of pageable pages below 16Mb that are good candidates for residence above Mb.

For entry point IEABRSMS: IEAVGFA, IEAVPUTD and IEAVRELA to suspend a unit of work and remember its identity.

For entry point IEAVS1BK: IEAVGFA to back out a failed stage 1 swap-in request.

Exit to:

For entry point IEAVGFD2: IEAVEDS0.

For all other entry points: Caller.

IEAVGFRR — Functional Recovery Routine for GETMAIN/FREEMAIN

Operation: This module recovers the function of GETMAIN/FREEMAIN after a machine or system error occurs.

Entry from: IEAVTRTS.

Exit to: IEAVTRTS.

IEAVGM00 — GETMAIN/FREEMAIN Service Routine

Operation: This module allocates (GETMAIN) and deallocates (FREEMAIN) virtual storage in the SWA and CSA and in the LSQA, SWA, and user region of each address space. A secondary function in support of system management facilities (SMF) is to produce actual storage-used values.

Entries:

CBBRANCH — for register type GETMAIN or FREEMAIN of a 16 byte cell from VSM control block cell pool.

GMBRANCH — for storage type GETMAIN.

FMBRANCH — for storage type FREEMAIN.

GLBRANCH — for register type GETMAIN or FREEMAIN of global subpools without local lock.

IEAVGMB — obtains storage for page tables.

IEAVGME — expands a VSM cell pool.

IEAVGMFB — frees space for page tables.

GSMFCRE — updates SMF accounting fields.

IEAVGMWP — puts the system in a wait when storage is not available for a global cell pool.

Entry from:

For entry point IGC004: IEAVESVC for storage type GETMAIN.

For entry point IGC005: IEAVESVC for register type, unconditional GETMAIN or FREEMAIN.

For entry point GETMAING: IEAVPRT0.

For entry point MRELEASF: IEAVPRT0.

For entry point MRELEASR: IEAVRELS.

For entry points IEAVGMSA, IEAVGMSU, IEAVGMGC, and IEAVGMNS: IEABGM03 to continue GETMAIN processing.

For entry point IEAVGMFC: IEAVGM03 to continue FREEMAIN processing.

For entry point IEAVGMGE: IEAVGM03 and IEAVGM04 to process GETMAIN/FREEMAIN errors.

Exit to: Caller.

IEAVGM03 — GETMAIN/FREEMAIN Service Routine

Operation: This module allocates (GETMAIN) and deallocates (FREEMAIN) virtual storage in the SWA and authorized user region of each address space.

Entries:

RMBRANCH — for register type, unconditional GETMAIN or FREEMAIN.

CRBRANCH — for register type GETMAIN or FREEMAIN branch entry (conditional or unconditional).

Entry from:

For entry point IGC010: IEAVESVC for register type, unconditional GETMAIN or FREEMAIN.

For entry point IGC120: IEAVESVC for register type GETMAIN or FREEMAIN.

Exit to: Caller.

IEAVGM04 — GETMAIN/FREEMAIN Service Routine

Operation: This module allocates (GETMAIN) and deallocates (FREEMAIN) virtual storage in the user region of each address space.

Entries:

IEAVGM4G — for GETMAIN processing.

IEAVGM4F — for FREEMAIN processing.

Exit to: Caller.

IEAVGPRR — GETPART/FREEPART Functional Recovery Routine

Operation: This module attempts to cleanup and recover from the following error situations encountered in module IEAVPRT0; program checks, machine checks, ABENDs, percolation from a subordinate FRR, and operator restart.

Entry from:

For entry point IEAVGPRR: IEAVTRTS.

For entry point PRTOERTN: IEAVSY50 via IEAVEDS0.

Exit to: IEAVEDS0.

For entry point IEAVGPRR: IEAVTRTS.

For entry point PRTOERTN: IEAVEDS0.

IEAVGTCL — GETCELL Routine

Operation: This module allocates a cell from an existing quickcell pool.

Entry from: A routine that requires quickcell allocation.

Exit to: Caller.

IEAVH600 — WQE Service Routine

This module executes in the communication task address space and provides the following cross memory services to manipulate write-to-operator queue elements (WQEs) for users of the WTO/WTOR and DOM macros.

- Locates a specific WQE in the communication task address space and copies it to the caller's WQE
- Copies the caller's WQE(s) to the communication task address space
- Sets the suspend indicator in the WQE for multi-line WTO processing
- Chains a WQE to an operator request element (ORE) for WTOR processing

Entry from: IEAVMWTO, IEAVVWTO, IEAVXDOM via a PC.

Exit to: Caller via PT.

Called routines: IEAVQ700.

Entry point: FRRCLNUP

At this entry point IEAVH600 performs cleanup for the communication task functional recovery routine (IEAVMFRR). If IEAVH600 had been processing a single-line WTO or WTOR when the error occurred, FRRCLNUP frees the WQE and adjusts the WQE counts in the UCM base. If EAAVH600 had been processing a multi-line WTO, FRRCLNUP turns off the suspend bit in the WQE and returns to IEAVMFRR with the number of WQEs processed.

Entry from: RTM.

Exit to: RTM.

Called routines: None.

IEAVID00 — Identify Routine

Operation: This module performs two functions:

- It identifies an embedded entry to the system.
- It identifies a single copy module loaded in subpool 0 to the system.

Entry from:

For entry point IGC041: SVC interruption handler.

For entry point FRRSVC41: RTM.

Exit to: EXIT prolog.

IEAVINV — Page Invalidation Routine

Operation: The routine is divided into two parts known as the master and the slave subroutines.

The master synchronizes all online processors, sets the invalid bit in the PGTE, purges the translation look-aside buffer (TLB) on the master processor, and signals the other processors to purge their TLBs. The slave portion purges the TLB on its processor when the master signals it to do so.

Entry from:

For entry point IEAVINV: An RSM routine.

For entry point IEAVINVA: IEAVEES.

Exit to: Caller.

IEAVIOCP — Paging I/O Completion Processor

Operation: This module performs the address-space-dependent processing required to indicate that a page-in operation has completed. It processes PCBs with I/O complete and PCBs for the address space in which IEAVIOCP is running.

Entry point: IEAVIOCP from IEAVEDSO.

Entry point: IEAVCPBR.

Exit to: Caller.

IEAVITAS — Initialize Address Space

Operation: This module builds and initializes RSM control blocks required to define an address space.

Entry from: IEAVGCAS.

Exit to: IEAVGCAS.

IEAVLK00 — BLDL/FETCH Interfaces

Operation: This module contains the description and code for several service routines:

LINK SVC 6 at entry point IGC006
XCTL SVC 7 at entry point IGC007
LOAD SVC 8 at entry point IGC008
DELETE SVC 9 at entry point IGC009
SYNCH SVC 12 at entry point IGC012
LINK SVC 122 at entry point IGC0X6
XCTL SVC 122 at entry point IGC0X7
LOAD SVC 122 at entry point IGC0X8

In addition to the above service routines, this module includes other code whose functions are described below.

For entry point IGC006 and IGC0X6:

- Performs a validity check of resources
- Establishes linkage to the module
- Searches for and determines the load module to which linkage is desired
- Creates and updates a program request block, an extended control block (XSB), and a contents directory entry under certain conditions
- Performs a sharing of load modules in accordance with reusability attributes
- Invokes the test module via an SVC call in IEAVLK01 for TSO modules in test

For entry point IGC007 and IGC0X7:

- Performs linkage to the load module specified
- Searches lists and libraries for the specified load module
- Creates and updates a PRB, an XSB, and a CDE as for IGC006
- Fetches the module

- Reclaims the current SVRB for type 4 SVC's during the processing

For entry point IGC008 and IGC0X8:

- Acquires a specified load module and retains the module for use by the task issuing the load
- Searches as in IGC006
- Creates a twelve-byte load list element (LLE) that points to the CDE for the requested module
- Increases the use count in the CDE and LLE when a module is loaded
- Creates a 32-byte CDE when none exists for the requested module
- Keeps a responsibility count in the LLE
- Returns entry point even if module is in use; will not queue load requests for a serially reusable module
- Causes modules to be fetched as for IGC006

For entry point IGC009:

- Indicates that the usage of a module is complete
- Locates the named module
- Decreases the use count in the CDE and LLE and if the count in the CDE goes to zero, it forces the purging of the module

For entry point IGC012:

- Allows the supervisor routine to give control to a problem program routine and to regain control in supervisor mode
- Gives task recovery the ability to schedule a synchronous exit routine
- Allows a routine which has received control via a SYNCH request to get control in a specified key
- Restores the caller's registers on return from a SYNCH request
- Provides options which allow the caller to set the storage key and the program key mask for the exit that is to receive control

For entry point IEAQCS01: This entry provides supervisor-assisted linkage for ATTACH requests.

For entry point IEAQCS02: This entry starts a queued request through mainline LINK.

For entry point IEAQCS03: This entry restarts a queued serially reusable request.

For entry point IEAVVMSR: This entry searches the pageable LPA directory for an LPDE representing the requested module.

For entry point IEAQCDSR: This entry searches a CDE queue for a requested name.

For entry point IEAQCS04: This entry address is the start of a list of addresses which IEAVLK01 uses to return to IEAVLK00. There is no linkage because this is an external reference and not executable.

For entry point CDLKBASE: This entry address resets the base address for IEAVLK00 on return from IEAVLK01. There is no linkage as this is an external reference and not executable.

Entry from:

For entry points IGC006 and IGC0X6: SVC IH.

For entry points IGC007 and IGC0X7: SVC IH.

For entry points IGC008 and IGC0X8: SVC IH.

For entry point IGC009: SVC IH.

For entry point IGC012: SVC IH.

For entry point IEAQCS01: ATTACH (IEAVEAT0) or memory create (IEAVEMCR).

For entry point IEAQCS02: POST (IEAVSY50).

For entry point IEAQCS03: POST (IEAVSY50).

For entry point IEAVVMSR: Control program and user routines.

For entry point IEAQCDSR: Control program and user routines.

Exit to:

For entry points IGC006 and IGC0X6: Routine to be linked via a branch entry to EXIT prolog; wait if request was queued.

For entry points IGC007 and IGC0X7: Routine to be linked via a branch entry to EXIT prolog; wait if the requested module was queued.

For entry point IGC008 and IGC0X8: Caller via EXIT prolog (IEAVEEXP).

For entry point IGC009: Caller.

For entry point IGC012: Caller using EXIT prolog.

For entry point IEAQCS02: Same as IGC006.

For entry point IEAQSC03: Same as IGC006.

For entry point IEAVVMSR: Return on register 14+0, if an LPDE is found; return on register 14+4 if no LPDE is found.

For entry point IEAQCDSR: Return on register 14+0, if a CDE is found; return on register 14+4 if no CDE is found.

Error exit: RTM via an ABEND macro.

IEAVLK01 — Program Management Search — Fetch Interface Module

Operation: This module performs the following services for IEAVLK00:

- Controls the order of queue and library searches
- Interfaces with the BLDL service routine using SVC 18
- Interfaces with the program fetch service routine (IEEWMSEPT) using a branch
- Provides an interface to test using SVC 61
- Processes alias PDS entries so that the major name module can be reused, if possible, for an alias request

Entry from: IEAVLK00 via branch.

Exit to: IEAVLK00.

IEAVLK02 — Program Management Resource Management Routines

Operation: This module cleans up CDEs and any associated storage at EXIT, DELETE, ABEND and EOT.

Entry from: ABEND.

Exit to: Program management.

Entry point CDHKEEP: This entry releases the module, extent list, major CDE, and associated minors as required.

Entry from: DELETE service routine (IGC009).

Exit to: Caller via BR 14.

Entry point GXLHKEEP: This routine frees CSA storage associated with modules that have been loaded into the CSA.

Entry from: RTM2 via a branch.

Exit to: Caller via a branch.

Entry point IEAPPGMX: This routine frees the program management resources when an RB exits and determines if a task termination is in progress, and if so, frees any LLEs left on the TCBLLE queue, their associated CDEs and its associated storage.

Entry from: EXIT (IEAVEOR) via branch with LOCAL lock held.

Exit to: Caller (SVC 3) via branch.

Entry point IEAPPGMA: This routine removes partially loaded programs during ABEND processing.

Entry from: IEAVTRT1 via branch.

Exit to: Caller via BR 14.

IEAVLK03 — Program Management Recovery Routine

Operation: This module attempts to prevent unwanted abends in the program manager by intercepting them and correcting errors if possible. The routine verifies CDE and LLE queues for abend situations that are not due to user error.

Entry point CDEVER: This entry point receives as input an address and verifies that the address points to a CDE.

Entry from: IEAVEQV0.

Exit to: Caller.

Entry point FRRPGMMG: This entry point provides program manager recovery for the basic program management functions.

Entry from: RTM.

Exit to: RTM.

Entry point FRRPGMX: This entry point is the recovery routine for the exit resoruce manager (IEAPPGMX). It verifies the CDE and LLE queues and attempts a retry.

Entry from: RTM.

Exit to: RTM.

IEAVMASV — Communication Task Asynchronous Service Routine

Operation: The asynchronous service routine performs TPUT services for the communication task. The TPUT services issue the WAIT macro instruction.

Entry from: IEAVTPUT and IEAVMASV.

Exit to: The dispatcher via a BR 14.

IEAVMDOM — Communication Task DOM Processor

Operation: The communication task DOM processing routine processes the DOM control blocks (DOMCs) created by the DOM macro instruction processing routine (IEAVXDOM). For each DOMC message ID, this routine attempts to eliminate the WQE that has the same message ID and the ORE associated with that WQE, if an ORE exists. If the WQE was retained in the action message retention buffer, it is removed. The device support processor is then called to eliminate graphic messages with the same message ID.

Entry from: IEAVMQWR.

Exit to: The communication task's wait service routine via a BR 14 instruction.

IEAVMDSV — Communication Task Device Service Routine

Operation: The device service routine prepares the communication task for and issues SVC 72 (the device support processor (DSP)). The preparation includes handling pending open, close, I/O completion interrupt, and attention interrupt processing. This routine also cleans up the write queue elements (WQEs), operator reply elements (OREs), and console queue elements (CQEs) when these control blocks are no longer needed, and assures that all messages have been processed for master tracing. Additionally, it retains messages in the action message retention buffer.

Entry from: IEAVMQWR and IEAVMDSV.

Exit to: Caller via a BR 14 instruction.

IEAVMED2 — WQE/ORE Purge Routine

Operation: This module purges WTO elements and ORE (operator reply elements) at task or address space termination. In particular, it:

- Frees any write wait blocks (WWBs) from the ORE and WQE chain.
- Marks any OREs to indicate that deletion is in progress.
- If this is an address space deletion, creates a DOMCB for a DOM by address space identifier (ASID).
- If deleting a job step, it creates a DOMCB for a DOM by ASID and jobstep TCB.
- Marks any active graphic device UCME to indicate that there is DOM work to be done.
- Stops monitor requests for this task or storage by calling the monitor control routine (IEAVMNTR).
- Posts the UCMDECB.
- Passes a task-related DOMCB to the subsystem interface.

Entry from: Address space or task termination.

Exit to: Caller.

IEAVMFIH — MSSFCALL SLIH Routine (IEAVMFIH)

Operation: IEAVMFIH processes class 2401 MSSF external interrupts, which are generated by the MSSF hardware when it finishes processing an MSSFCALL DIAGNOSE instruction.

Entry from: IEAVEEXT.

Exit to: Caller.

IEAVMFRM — MSSFCALL Resource Manager

Operation: IEAVMFRM cleans up any MSSFCALL resources held by a terminating task or address space. These MSSFCALL resources include the MSFAB and MSFCB control blocks and the MSSFCALL TP PORT serialization resource.

Entry from: IEAVTMMT during address space termination or IEAVTSKT during task termination.

Exit to: Caller.

IEAVMFRR — Communication Task Functional Recovery Routine (FRR) or ESTAE Controller

Operation: The FRR/ESTAE controller receives control from recovery termination management (RTM). The controller processes both FRR and ESTAE attempts to recover from abnormal terminations in the communication task.

Entry from: RTM via FRR stack.

Exit to: RTM via a BR 14 instruction.

IEAVMNTR — Monitor Queue Manager

Operation: This module maintains the monitoring status of operators' consoles and TSO terminals. This is done primarily for the MONITOR and STOPMN command processors.

Entry from: IEAVMED2, IEAVSWCH, IEE5503D, IEE7103D, IKJEFF00, or IEE4603D.

Exit to: Caller.

IEAVMODE — Mode Change Routine

Operation: This module changes the key and state in the SVC old PSW (SVCOPSW), changes the system mask, or changes the PSW key mask (control register 3). It changes the state of the system by altering the RB old PSW.

Entry from: SVC IH.

Exit to: EXIT prolog.

IEAVMQR0 — Communication Task Process Message for Inactive Terminal (Console)

Operation: This routine gives the master console operator an opportunity to delete, queue to the master console, or continue queueing messages that are sent to a specific console by identification.

Entry from: IEAVMWSV.

Exit to: The dispatcher via a BR 14 instruction.

IEAVMQWR — Communication Task Wait Service Routine

Operation: The wait service routine waits on all of the event control blocks (ECBs) that are posted for the communication task.

Entry from: IEAVN701.

Exit to: No exit; this is a never-ending routine.

IEAVMSF — Monitoring and System Support Facility (MSSFCALL) SVC Routine

Operation: IEAVMSF provides the interface between MVS software and the hardware's monitoring and system support facility (MSSF). It determines whether the requested MSSFCALL command can be processed, and, if so, issues an MSSFCALL DIAGNOSE instruction on behalf of the requestor.

Entry from: SVC FLIH (IEAVESVC).

Exit to: Address in register 14.

Called routines: Validity check routine (IEAVEVAL).

IEAVMSFS — MSSFCALL SRB Routine

Operation: After the MSSF hardware finishes executing an MSSFCALL DIAGNOSE instruction that IEAVMSF issued while processing an MSSFCALL

SVC, it generates a class 2401 MSSF external interrupt. IEAVMFIH receives control to schedule IEAVMSFS, which posts the ECB that IEAVMSF is waiting on.

Scheduled by: IEAVMFIH.

IEAVMWSV — Communication Task Console Queueing Routine

Operation: The console queueing routine selectively queues messages to console output queues and passes eligible messages for master tracing.

Entry from: IEAVMDSV, IEAVMQWR.

Exit to: The device service routine (IEAVMDSV), which returns control to the communication task's wait service routine (IEAVMQWR).

IEAVMWTO — Communication Task Multiple-Line WTO Service Routine

Operation: The communication task multiple-line WTO (MLWTO) service routine prepares major and minor write queue elements (WQEs) associated with multiple line messages.

Entry from: IEAVVWTO.

Exit to: IEAVVWTO.

Called routines: IEAVH600, IEAVM700, IEFJSREQ.

IEAVM700 — MPF Scan Routine

Operation: This module scans the MPF table built by IEECB805. The table contains message IDs and prefixes of messages to be suppressed. IEAVM700 determines whether the message ID or prefix passed as input matches any of the entries in the MPF table.

Entry from: IATCNMR, IEAVVWTO, IEAVMWTO.

Exit to: IATCNMR, IEAVVWTO, IEAVMWTO.

IEAVOUT — Page Out

Operation: This module processes all page-out requests initiated by the PGOUT macro instruction. If the page needs to be written to auxiliary storage, this module starts the I/O request.

Entry from: IEAVPSI.

Exit to: IEAVPSI.

IEAVPCB — PCB Manager

Operation: This module moves PCBs to various queues as requested and ensures that an adequate supply of PCBs is in the free queue pool for requesters. It also maintains the SRB pool.

Entry from: An RSM module or IEAVNIP0.

Exit to: Caller.

IEAVPFTE — PFTE Enqueue/Dequeue Routine

Operation: This module moves PFTEs to and from PFTE queues as directed or needed. It also restarts deferred general frame allocation (GFA) requests and monitors the level of the AFQ (available frame queue).

Entry from: RSM modules.

Exit to: Caller.

IEAVPIOI — Swap I/O Initiator

Operation: This module invokes the ASM interface routine to initiate paging I/O for swap-out of an address space.

Entry from: IEAVEDS0.

Exit to: IEAVEDS0.

IEAVPIOP — Page I/O Post

Operation: This module performs functions relating to paging I/O, including frame validation and freeing of PCBs and frames.

Entry from:

For entry point IEAVPIOP: ILRI0C00, ILRMON00, or ILRQIO00.

For entry point IEAVOPBR: IEAVIOCP, IEAVTERM, IEAVRELS, IEAVAMSI, IEAVGFA, or IEAVSOUT.

Exit to: Caller.

IEAVPIX — Program Check Interrupt Extension

Operation: This module functions as an interface routine between the program check interruption handler (PCIH) and the general frame allocation (GFA) routines.

Entry from: IEAVEPC.

Exit to: IEAVEPC.

IEAVPREF — Frame Allocation Steal Function

Operation: This module steals a frame from a local frame queue, the common frame queue, or the real storage buffer queue to be used for SQA, LSQA, or for a fixed page.

Entry from: IEAVGFA or IEAVSQA.

Exit to: Caller.

IEAVPRT0 — GETPART/FREEPART Routine

Operation: This module allocates or frees a virtual region.

Entry from: IEAVGM00.

Exit to: IEAVGM00 or IEAVEQR.

Error exit: IEAVTRT2.

IEAVPSI — Page Services Interface (PSI) Module

Operation: This module receives all page service requests. The requests serviced are PGFIX, PGFREE, PGLOAD, PGRLSE and PGOUT. This module checks input parameters for errors and routes control between the page services modules necessary to satisfy the input request.

Entry from:

For entry points IGC112 and IGC113: IEAVESVC.

For entry points IEAVPSIB, IEAVPSIF, IEAVPSIX, IEAVPSIY, IEAVPSIZ, and IEAVPSXM: A supervisor state key 0 routine.

For entry points IEAVPSII and NEXTVSL: An RSM routine.

Exit to:

For entry points IGC112 and IGC113: IEAVEOR.

For all other entry points: Caller.

Error exit:

For entry points IGC112 and IGC113: IEAVRCV.

For entry points IEAVPSIF, IEAVPSIX, IEAVPSIY, IEAVPSIZ, and IEAVPSXM: IEAVDSPC.

IEAVQ700 — Communications Task Queue Scanner.

Operation: This module scans a particular communications task resource queue to search for an entry containing those characteristics specified by the caller. The address is returned to the caller, when the entry is found. The address field is set to zero when no entry is found.

Entry from: IEAVH600, IEAVMDOM, IEECB804, IEE8103D.

Exit to: Caller.

IEAVRCF — Real Storage Reconfiguration Routine

Operation: This module attempts hardware recovery (storage data and key errors), fields requests for varying storage offline and online, and provides current frame status.

Entry from:

For entry point IEAVRCF: IEEMPVST.

For entry point IEAVRCFC: IEAVRCF via IEAVEDS0.

For entry point IEAVRCFI: IEAVPFTE.

Exit to: Caller.

IEAVRCF3 — Real Storage Reconfiguration Module

Operation: This module adds to or subtracts from the real storage frames currently available for use by the system. This module also records the status of the real frames.

Entry from:

For entry points IEAVRCF3 and IEAVPBAD: IEEVSTGL.

For entry point IEAVRF3I: IEAVPFTE.

For entry point IEAVRF3C: Dispatcher.

Exit to: Caller.

Called routines: IEAVPFTE and IEAVPCB.

IEAVRCV — Real Storage Management Recovery

Operation: This module attempts to recovery from program checks, machine checks, operator restarts, percolation from subordinate FRRs, and expected and unexpected ABEND requests in RSM routines.

Entry from:

For entry point IEAVRCV: IEAVTRTS.

For entry point IEAVRCV2: IEAVTRTS to handle errors is IEAVRCV.

Exit to: Caller.

IEAVRELS — Page Release (PGRLSE)

Operation: This module frees real and auxiliary storage associated with a virtual page. In the process, it quiesces any outstanding paging I/O for the page.

Entry from:

For entry point IEAVRELS: IEAVPSI.

For entry point IEAVRELV: IEAVGM00.

Exit to: Caller.

Entry point: IEAVRELF from IEAVIOCP, IEAVFREE, or IEAVTERM.

Exit to: Caller.

IEAVRFR — Real Frame Replacement and UIC Update

Operation: This module is invoked by SRM when the steal algorithm is to be executed. It is passed a criteria number by which it is to decide which frames are eligible to be stolen.

This routine is used to remove pages from a given address space and return the frames to the available pool.

The unreferenced interval count (UIC) update routines (IEAVRFRP for private and IEAVRFRC for common) are invoked by SRM to update the usage history and the unreferenced interval count for each PFTE on a given frame queue.

Entry point: IEAVRFR from IRARMSRV.

Exit to: IRARMSRV.

Entry from: At IEAVRFRP from IRARMSRV.

Exit to: IRARMSRV.

Entry from: At IEAVRFRC from IRARMSRV.

Exit to: IRARMSRV.

IEAVRTI0 — Timer Second Level Interrupt Handler

Operation: This module processes interrupts caused by a synchronization check, by the processor timer, or by the clock comparator. This module also contains the TQE enqueue and dequeue routines.

Entry point: IEAVEEXT at IEA0TI00.

Exit to: IEAVEEXT.

Entry point: IEAVRT00, IEAVRTI0, IRARMSRV, IEAVRTOD, IEE6503D, IEAVRT02, IGX00013 at IEAQTE00.

Exit to: Caller.

Entry point: IEAVRT00, IEAVRTI0, IRARMSRV, IEAVRT11, IEE6503D at IEAQTD00.

Exit to: Caller.

Entry point: IEAVRTI0, IEAVRTOD at IEAVRCKQ.

Exit to: Caller.

Entry point: IEAVRQCK from IEAVEXS.

Exit to: IEAVEXS.

Entry point: IEAVRSAE from IEAVEDS0.

Exit to: IEAVEDS0.

IEAVRTI1 — Timer Purge and Recovery

Operation: This module performs four functions: the TQE purge function called by a terminating task or memory; the hardware recovery routine called by RTM; initialization timer control block fields and timer hardware when a new processor is varied online; and the FRR for timer supervision which includes a TQE validation routine.

Entry point: IEAQPGTM from IEAVTSKT or IEAVTMMT.

Entry point: IEAVRSPG from IEAVEDS0.

Entry point: IEAVRCLS from IEAVTRTH.

Entry point: IEAVRCLX from IEAVEDS0.

Entry point: IEAVRNEW from IEEVCPU.

Entry point: IEAVRSPN from IEAVEDS0.

Entry point: IEAVRFRR from IEAVTRTS.

Entry point: IEAVRTVR from IEAVRFRR or IEAVEQV0.

Exit to: Caller.

IEAVRTOD — TOD Clock Manager

Operation: This module initializes TOD clocks at IPL and when a processor is varied online and resets or resynchronizes TOD clocks suffering machine checks.

Entry point: IEAVRINT from IEEVIPL.

Entry point: IEAVRSSC from IEAVRTOD and IEEVCPU.

Entry point: IEAVRCLA from IEAVEDS0.

Entry point: IEAVRNOT from IEEVCPU.

Entry point: IEAVRCAN from IEEVCPU.

Exit to: Caller.

IEAVRT00 — TTIMER Service Routine (SVC 46) and STIMER Service Routine (SVC 47)

Operation: TTIMER will supply the time remaining in a task's time interval which was previously established via the STIMER routine. TTIMER will also, if requested, cancel the interval. STIMER establishes a time interval for the requesting task.

Entry point: IGC0004F from IEAVESVC.

Entry point: IGC0004G from IEAVESVC.

Exit to: IEAVEOR.

Error exit: IEAVTRT2.

IEAVRT01 — TIME Service Routine (SVC 11)

Operation: This module supplies the user with local time and date, or optionally, with Greenwich Mean Time and date. The current value of the Time-of-Day clock can also be requested.

Entry point: IGC0001A from IEAVESVC.

Exit to: IEAVEOR.

Entry point: IEAVRTME from IEAVEES.

Exit to: IEAVEES.

Entry point: TIMESTAE from IEAVTAS1.

Error exit: IEAVTRT2.

IEAVRT02 — SETDIE Routine

Operation: This module allows a system program to establish a real-time interval. After the interval has passed, the program's disabled interrupt exit (DIE) routine gets control.

Entry point: IEAVRDIE from system caller.

Exit to: System caller.

IEAVRT03 — SRB Time Limit Initialization

Operation: This module establishes a specified time limit for a program executing in SRB mode.

Entry from: Issuer of a SRBTIMER macro.

Exit to: Caller.

Error exit: Return to caller that is not in SRB mode.

Called routines: Dispatcher job step timing collection subroutine (entry point DSJSTCSR in IEAVEDS0).

IEAVSETS — STATUS Service Routine

Operation: The module manipulates TCB indicators to change the dispatchability of tasks, SRBs, or the system.

For entry point IGC079: Manipulates dispatchability of tasks, SRBs, or the system.

For entry point IGC07902: Manipulates the dispatchability of tasks and SRBs.

For entry point IGC07903: Starts SRB in swapped-in address spaces.

For entry point IGC07904: Resumes the processing which had been suspended pending the release of a CML lock.

For entry point IEAVSSNQ: Stops non-quiescable SRBs in address spaces being swapped-out.

For entry point IEAVESSS: Completes stop SYNCH processing.

For entry point IEATRSCN: Searches the subtask tree structure.

For entry point IEAVSET1: Clears all other processors of active tasks dispatched in the currently dispatched address space.

Entry from:

For entry point IGC079: SVC IH.

For entry point IGC07902: Supervisor routines via branch entry.

For entry point IGC07903: Swap-in.

For entry point IGC07904: The lock manager.

For entry point IEAVSSNQ Swap-out.

For entry point IEAVESSS: EXIT and EXIT prolog.

For entry point IEATRSCN: Branch entry from system routines.

For entry point IEAVSET1: The attention exit scheduling routine or other system routines.

Exit to:

For entry point IGC079: Exit prolog.

For entry point IGC07902: Caller or dispatcher.

For entry point IGC07903: Caller or dispatcher.

For entry point IGC07904: Caller or dispatcher.

For entry point IEAVSSNQ: Caller or dispatcher.

For entry point IEAVESSS: Caller or dispatcher.

For entry point IEATRSCN: Branch to caller via BR 11 if no task found, or BR 14 if task is found.

For entry point IEAVSET1: Caller.

IEAVSOUT — Swap Out

Operation: This routine proposes to process or move an address space from the system at the direction of the system resource manager (SRM). This includes purging all paging I/O for the address space, purging fixes for the address space, freeing LSQA frames, indicating which pages should be swapped in, and building the control blocks which will control the swap-out of pages which need to be written to auxiliary storage.

Entry point: IEAVSOUT from IEAVAR02.

Exit to: IEAVAR02.

IEAVSQA — LSQA or SQA Allocation

Operation: This module allocates real storage frames for use as SQA or LSQA pages.

Entry point: IEAVSQA from IEAVGM00 or IEAVITAS.

Exit to: Caller.

IEAVSTAA: Communication Task ESTAE Routine

Operation: This ESTAE routine is the last attempt to recover from an abnormal termination in the communication task. Previous attempts to recover may have been made by the functional recovery routine (FRR) and ESTAE processing (IEAVMFRR). IEAVSTAA reinitializes all fields that are necessary to make the system believe the communication task has not failed.

Entry from: RTM.

Exit to: The dispatcher to end the communications task for later reinitialization.

Error exit: A branch return to the recovery termination management routine (RTM) based on the address in register 14.

IEAVSTA0 — (E)STAE Service Routine (SVC 60)

Operation: This module creates, overlays, cancels, or propagates STAE control block(s).

Entry from: SVC 60 call or a branch entry by another SVC routine.

Exit to: Caller via exit if SVC entered, or directly to caller if branch entered.

Exit-error: ABEND caller.

IEAVSWCH — Communication Task Console Switch Routine

Operation: The console switch routine analyzes and switches console functions.

Entry from: IEAVMWSV, IEECVETW, IEAVVCTR, IEAV1052, IEECVETC, IEAV1443, IEAV2540, IEE8B03D.

Exit to: A branch return to the address in register 14.

IEAVSWIN — Swap In Processor

Operation: This module performs the processing required to initiate a swap-in of an address space from the auxiliary paging space. The secondary

entry performs processing to connect the address space to the set of active address spaces when the required I/O is complete.

Entry point: IEAVSWIN from IEAVEDS0.

Exit to: IEAVEDS0.

Entry point: IEAVSIRT from IEAVPIOP, IEAVGFA, IEAVIOCP or IEAVTERM.

Exit to: Caller.

IEAVSWPC — Swap-out Completion Processor

Operation: This routine frees the frames and PCBs for swap-out and removes the address spaces from the dispatching queue.

Entry from: ILRPAGCM.

Exit to: Caller.

IEAVSWPP — Swap-In Completion Processor

Operation: Scheduled by Swap-In Root Exit (IEAVSIRT entry point in IEAVSWIN). This module starts input page operations for stage 2 pageable working set pages and posts the region control task (RCT) to restart task-mode processing in a swapped-in address space.

Entry from: IEAVEDS0.

Exit to: IEAVEDS0.

IEAVSY50 — Wait/Post Service Routine

Operation: This module performs two services:

- WAIT
  - Which causes the caller to stop processing and wait for the occurrence of one or more events.
- POST
  - Which signals completion of an event in an ECB and determines if the waiting task has become ready.
  - It also identifies and deletes user POST exit routines and routes control to them.

  For entry IEA0PT02: Performs the POST service.

  For entry IEA0PT03: Performs the POST service for programs executing as a POST exit routine.

  For entry IEA0PT0E: Performs the POST exit identification and deletion service.

  For entry IEARPOST: POST resource manager.

Entry from:

For entry IGC001: SVC IH.

For entry IEAVWAIT: Supervisor routine via branch entry.

For entry IGC002: SVC IH.

For entry IEA0PT01: Supervisor routine via branch entry.

For entry IEA0PT02: Supervisor routine via branch entry.

For entry IEARPOST: RTM.

For entry IEA0PT03: Supervisor routine via branch entry.

For entry IEA0PT0E: Supervisor routine via branch entry.

Exit to:

For entry IGC001: Exit prolog (IEAVEXP1), or ABEND.

For entry IEAVWAIT: Dispatcher (IEA0DS).

For entry IGC002: Exit prolog or ABEND.

For entry IEA0PT01: Caller.

For entry IEA0PT02: Caller.

For entry IEARPOST: Caller.

For entry IEA0PT03: Caller.

For entry IEA0PT0E: Caller.

IEAVTABD — ABDUMP

Operation: This module manages the SYSABEND, SYSUDUMP, and SYSMDUMP data sets. It provides the interface to SNAP and SVC dump for dump requests to one of these data sets during recovery/termination processing.

Entry from: IEAVTRTC via BALR.

Exit to: IEAVTRTC.

IEAVTABI — ABDUMP Initialization

Operation: This module reads PARMLIB IEAABD00, IEADMP00, and IEADMR00 and sets the requested dump options in the recovery termination control table (RTCT) for later use by ABDUMP (IEAVTABD).

Entry from: IEAVNPA6 via BALR.

Exit to: IEAVNPA6.

IEAVTACR — Alternate processor Recovery

Operation: This module uses a 'good' (recovery) processor to take over the work in progress on a 'dead' (malfunctioning) processor at the time of its failure, and to supervise the cleanup of that work until the system can resume normal operation:

Entry from:

- RMS EMS/MFA SLIH on the recovery processor.
- Lock manager and certain global spinners when conflict is detected between recovery and dead processor work.
- Dispatcher when recovery or dead processor work is interrupted.

Exit to:

- Caller via register 2.
- Interrupted work of other processor. If this is first time for dead processor work exit to RTM for FRR recovery.
- If work of each processor is in enabled condition exit to dispatcher.

IEAVTADR — Address Conversion Routine

Operation: This module converts a coded string representing a direct or indirect address into a four-byte virtual address.

Entry from: IEAVTSLP via BALR, AHLTSLIP via BALR, IEECB905 via BALR.

Exit to: Caller.

IEAVTAS1 — Task Recovery Pre-exit Processing

Operation: This module sets up workareas for ESTA exit, gives the exit control and calls IEAVTAS2 and IEAVTAS3 for post-exit processing. The pre-exit functions of this module are as follows:

- Identify the SCB (ESTAE/STAE exit).

- Obtain and initialize SDWA.
- Perform the user I/O options requested on ESTAE macro.
- Set up interface for the user exit routine.
- Issue SYNCH macro to pass control to the user exit routine.

Entry from: IEAVTRTC via BALR.

Exit to: IEAVTRTC.

IEAVTAS2 — Task Recovery Post Exit Processing

Operation: This module performs the requests made by the user exit routine. Its functions are the following:

- Perform a validity check on the SDWA.
- Track SDWA by issuing HOOK macro.
- Perform recording if requested.
- Copy dump options from SDWA.
- Suppress STAI percolation, if requested.
- Determine whether to retry or continue with termination.

Entry from: IEAVTAS1 via BALR.

Exit to: IEAVTAS1.

IEAVTAS3 — Setup for User Retry or Continue with Termination

Operation: This module performs processing for user retry or percolation. Processing consists of the following:

Retry:

- Identify the RB for retry.
- Initialize or freemain SDWA for retry.
- Close embedded DCBs.
- Purge paging I/O.
- Purge outstanding WTOR requests for STAE retry only.
- Purge entries in type 1 message table.
- Purge enqueued resources if ESTAR.
- Purge TQEs if ESTAR.
- Set pointer to EXIT (RB) or exit prolog (SVRB) in resource PSW.
- Perform SCB clean-up.
- Handle pseudo retry into IEAVTAS1 if there is recursion.

Percolation: (continue with termination)

- Free SDWA

Entry from: IEAVTAS1 via BALR.

Exit to: IEAVTAS1.

IEAVTB00 — SPIE/EXTRACT Service Routine

Operation: The module provides two services:

- SPIE, which provides the problem program with a means of specifying an error exit routine in response to one or more program error interruptions.
- EXTRACT, which provides a means of obtaining the address of the values of a set of fields without going through the system.

For entry IGC0001D: Performs SPIE mainline processing.

For entry IEAVSPIE: SPIE resource manager.

For entry IEAVSPI: Checkpoint/restart entry.

For entry IGC00040: Performs EXTRACT mainline processing.

For entry IGC00040+8: Performs EXTRACT mainline processing.

Entry from:

For entry IGC0001D: SVC IH.

For entry IEAVSPIE: RTM.

For entry IEAVSPI: Checkpoint/restart.

For entry IGC00040+8: Supervisor routines via branch entry.

Exit to:

For entry IGC0001D: Exit prolog.

For entry IEAVSPIE: Caller.

For entry IEAVSPI: Caller.

For entry IGC00040: Exit prolog.

For entry IGC00040+8: Caller.

IEAVTERM — Paging Termination Services

Operation: This module quiesces paging I/O for a particular RB or TCB. For a task being terminated, RTM may also request that all pages fixed for a task be freed.

Entry point: IEAVTERM from IEAVTSKT.

Exit to: IEAVTSKT.

IEAVTEST — TESTAUTH Service Routine

Operation: This routine tests the authorization of any program to perform a specified function.

Entry from:

For entry IGC119: SVC IH.

For entry IEAVTEST: Supervisor routine via branch entry.

Exit to:

For entry IGC119: EXIT prolog.

For entry IEAVTEST: Caller.

IEAVTFHX — HEX Format Subroutine

Operation: This subroutine formats a single summary dump record in standard hexadecimal dump format with characters on the side.

Entry from: IEAVTFSD.

Exit to: Caller.

IEAVTFMT — MAINLINE for RTM Control Block Formatter

Operation: Locates the RTM control blocks associated with the TCB passed in the parameter list. It then calls the individual formatters for each RTM control block.

Entry from: IEAVAD01 or AMDPRUIM.

Exit to: Caller.

IEAVTFRD — Summary Dump Read Subroutine

Operation: This subroutine reconstructs the records of a summary dump from a dump data set being processed by AMDPRDMP.

Entry from: IEAVTFSD.

Exit to: Caller.

IEAVTFRT — Interface between IEAVTFSD and Several RTM Control Block Format Subroutines

Operation: Using summary dump data to simulate a total dump environment this module invokes the RTM control block format subroutines IEAFTIHS, IEAFTFRR, IEAFTRT2.

Entry from: IEAVTFSD.

Exit to: Caller.

IEAVTFSD — AMDPRDMP Service Aid Exit for SUMDUMP Control Statement

Operation: This module controls the formatting of an entire summary dump by calling format subroutines to process each summary dump record.

Entry from: AMDPRDMP via the user exit interface (entry point AMDUSRXT in AMDPRUIM).

Exit to: Caller.

IEAVTFTM — Message and Subroutine Name CSECT for IEAVTFSD

Operation: This message CSECT contains all the text for the dump output headers and comments generated by IEAVTFSD, IEAVTFRD, and IEAVTFHX. It also contains a branch table of subroutines to load and call for each type of summary dump record.

Used by: IEAVTFSD.

IEAVTGLB — SLIP Global PER Activation/Deactivation Routine

Operation: This module manages the global resources on all processors necessary to activate, deactivate, or adjust PER monitoring. This module is scheduled by IEECB905, IEAVTJBN, IEAVTLCL, and IEAVTSLP.

Entry from: SRB dispatcher.

Exit to: SRB dispatcher.

IEAVTJBN — SLIP PER Selection Interface Routine

Operation: This module serves as an interface between address space create and IEAVTLCL to determine if a new address space should have PER monitoring active. It also serves as the interface between scheduler routines (started task control and SWA create control) and SLIP PERa routines to determine if a LOGON, START, or job select requires a change to PER monitoring in an address space.

Entry from: IEAVEMRQ via BALR, IEESB605 via BALR, IEFIB600 via BALR.

Exit to: Caller.

IEAVTLCL — SLIP Local PER Activation/Deactivation Routine

Operation: This module determines if PER monitoring should be on or off for the address space in which it is running and adjusts the RB and ASCB indicators accordingly. IEAVTGLB also schedules this routine to adjust the RB and ASCB indicators as required. This routine is scheduled by IEAVTGLB and IEAVTJBN.

Entry from: SRB dispatcher.

Exit to: SRB dispatcher.

Called routines: IEATRSCN, IEAVESSE, IEAVSETS.

IEAVTMMT — Memory Termination Purges

Operation: This module handles the purging of resources held by a terminating address space.

Entry from: IEAVTRTE via BALR.

Exit to: Caller.

IEAVTMRM — RTM Memory Resource Manager

Operation: This module frees all SDWAs obtained from SQA by RTM1.

Entry from: IEAVTMMT via BALR.

Exit to: Caller.

IEAVTMSI — Recovery Termination's Master Scheduler Initialization Controller

Operation: This module creates the following three recovery termination tasks in the master address space:

- Recording task
- Address space termination control task
- SVC dump task

Entry from: Master scheduler (IEEMB860) via a LINK macro to begin recovery/termination initialization.

Exit to: Master scheduler (IEEMB860) via a RETURN statement with return code 0 or 4.

IEAVTMTC — Memory Termination Controller

Operation: This module controls the process of responding to a request for address space termination.

Entry from: IEAVTMSI via ATTACH.

Exit to: None. This is a never-ending task that waits on an ECB when there is no work to do.

IEAVTMTR — Memory Termination Requestor

Operation: This module is attached by the memory termination controller to invoke memory termination purges via SVC 13.

Entry from: IEAVTMTC via ATTACH.

Exit to: Caller.

IEAVTPER — Program FLIH/SLIP PER/Space Switch Interface Module

Operation: This module provides the interface used to call the SLIP processor (IEAVTSLP) to handle a PER interrupt or the space switch handler (IEAVTSSH) to handle a space switch interrupt detected by the program check FLIH (IEAVEPC).

Entry from: IEAVEPC via BALR.

Exit to: Caller.

IEAVTPMT — Type 1 Message Table Handler

Operation: This module prints or purges entries in the type 1 message table for the requested task.

Entry from:

IEAVTRTC via BALR for printing and purging.
IEAVTAS3 via BALR for purge.
IEAVTSKT via BALR for purging.

Exit to: Caller.

IEAVTRCE — Trace Routine

Operation: This module provides a trace function for the following events:

- External interruptions
- I/O interruptions
- Program interruptions
- SVC interruptions
- Start I/O (from IOS)
- Task dispatcher exit conditions
- Event dispatcher exit conditions

For entry point TREX (the trace entry for external interrupts): External interrupt handler.

For entry point TRIO (the trace entry for I/O interrupts): Supervisor.

For entry point TRPI (the trace entry for program interrupts): Program interrupt handler.

For entry point TRSVC (the trace entry for SVC interrupts): SVC interrupt handler.

For entry point TRDISP (the trace entry for task dispatch): Dispatcher.

For entry point TRSRB1 (the trace entry for event dispatch): Dispatcher.

For entry point TRSRB2 (the trace entry for event redispatch): Dispatcher.

For entry point TRSIO (the trace entry for START I/O): START I/O.

For entry point IEAVTRC0 (to turn system tracing on or off): GTF or SDUMP.

For entry point TRCEFRR (trace recovery routine): RTM.

Exit to: Caller by branching to the address in register 11.

IEAVTRER — Recording Request Routine

Operation: This module accepts requests for recording on SYS1.LOGREC or for writing to the operator, and it schedules the recording/writing to occur asynchronously under the recording task. Also, if this is an emergency request from the machine check handler, the termination routine's emergency recorder returns the address and the length of the scheduled but not yet written logrec request for recording.

Entry from: System routines via RECORD macro instruction.

Exit to: Caller.

IEAVTRET — Asynchronous Recording Task

Operation: This module writes recording requests.

Entry from: IEVTMSI via ATTACH, subsequently IEVTRER schedules an SRB to POST RTCTRECB.

Exit to: None. This is a never-ending task that waits on an ECB when there is no work to be done.

IEAVTRMC — CALLRTM TYPE=RMGRCML

Operation: At address space termination time, this module prepares for cleanup every SRB or task in the terminating address space that holds the local lock of another address space.

Entry from: IEAVLKRM via CALLRTM.

Exit to: IEAVLKRM.

IEAVTRML — Installation Resource Manager List

Operation: This CSECT can be replaced with a CSECT containing a list of modules to be called by task and address space termination. It is defined to look like a list with no entries.

Entry from: Not executable.

Exits: None.

IEAVTRS0 — RTM1 Service Routines

Operation: This module contains several entry points used by IEAVTRTM. Two of these entries (IEAVTRS2 and IEAVTRS4) are also called internally by IEAVTRS3. See the specific entry point for a description of its function.

Entry point IEAVTRS1: This entry point verifies that an ASID received by RTM1 as input represents a valid address space.

Entry from: IEAVTRTM via BALR.

Exit to: IEAVTRTM via BALR.

Entry point IEAVTRS2: At this entry point, the module updates an EED with the dump options passed by the invoker of the ABEND, CALLRTM, or SETRP macros.

Entry from: IEAVTRTM or IEAVTRS3 using BALR.

Exit to: IEAVTRTM or IEAVTRS3 using BALR.

Entry point IEAVTRS3: At this entry point, the module updates EEDs with error information and chains them on a queue.

Entry from: IEAVTRTM via BALR.

Exit to: IEAVTRTM via BALR.

Entry point IEAVTRS4: This entry point puts the ERRORID (sequence number, logical processor identification, and time stamp) and the FRR-to-ESTAE communication buffer into the EED.

Entry from: IEAVTRTM or IEAVTRS3 via BALR.

Exit to: IEAVTRTM or IEAVTRS3 using BALR.

Entry point IEAVTRS5: IEAVTRTM invokes this maintenance entry point when a user attempts to terminate an address space that cannot be terminated. This module uses the software recording facility to record the SDWA along with its variable recording area.

Entry from: IEAVTRTM using BALR.

Exit to: IEAVTRTM using BALR.

Called routines: IEAVTRER.

Entry point IEAVTRS6: IEAVTRTM uses this entry point to interface with the status routine to make a task dispatchable.

Entry from: IEAVTRTM using BALR.

Exit to: IEAVTRTM using BALR.

Entry point IEAVTRS7: At this entry point, the module interfaces with IGPTERM to terminate the system by issuing the IEA802W message, which indicates that the system is in an A00 wait state.

Entry from: IEAVTRTM via BALR.

Exit to: IGFPTERM via BALR.

IEAVTRTC — RTM2 Mainline Controller

Operation: This module controls the processing among subfunctions, including task recovery, dumping, and subtask handling.

Entry from: IEAVTRT2 via BALR.

Exit to: IEAVTRT2.

IEAVTRTE — Control Task Purges, Memory Purges and Exit

Operation: This module invokes modules to perform task and address space purges and causes the appropriate exit from RTM2.

Entry from: IEAVTRT2 or IEAVTSKT via BALR.

Exit to: PROLOG or to STATUS to set the task temporarily or permanently nondispatchable.

IEAVTRTH — RTM Hardware Error Processor

Operation: This module interfaces with hardware repair routines and records a MCH LOGREC buffer to SYS1.LOGREC for all CALLRTM TYPE=MACHCK entries. The module also accumulates data describing the error and its repair status for inclusion in the recovery routine interface.

Entry from: IEAVTRTM via BALR.

Exit to: Caller.

IEAVTRTM — Recovery Termination Manager Number 1

Operation: This module performs the following functions:

- Interfacing with the module IEAVTRTH for recording and repairing damage after a hardware failure

- Interfacing with the system recovery manager module IEAVTRTS for routing control to the recovery routines that protect a supervisor path and for recording software incidents
- Percolating errors to tasks or address spaces that are affected by the error
- Interfacing with the module IEAVTRTR which handles recovery for RTM1
- Handling the ABTERM, MEMTERM, and STERM services

Entry from: IEAVTRT1 via BALR.

Exit to: IEAVTRT1 via BALR, unless system termination is necessary, then exit to IEAVTRS0.

Called routines: IEAVTRS0, IEAVTSLP.

IEAVTRTR — RTM Recovery Routines

Operation: This module contains the recovery routines for the RTM1 routines and also routines to perform SLIP processing.

Entry point:

For entry FREEDCEL: This entry interfaces with the QUICKCELL routine to free any quickcells that are no longer needed.

For entry RTMRSFRR: This entry provides recovery for IEAVTRTM if a recursive error occurred during an attempt by RTM to perform its reschedule function.

For entry RTHFRR: This entry provides recovery for IEAVTRTH.

For entry RTMSMFRR: This entry continues the recovery initiated by RTMRSFRR.

For entry RCOVGETM: This entry defines an FRR that provides recovery for IEAVTRTS when a GETMAIN for an SDWA fails while processing an error in unlocked SRB mode or for an enabled unlocked task with FRRs.

For entry RCOVRCRD: This entry defines a FRR that provides recovery of IEAVTRTS in the event that recording an SDWA on behalf of an FRR fails.

For entry RCOVRGTF: This entry defines a FRR that provides recovery of IEAVTRTS in the event that GTF (invoked by IEAVTRTS to trace FRR recovery) should fail.

For entry RECVRRTM: This entry provides recovery for RTM1 during that part of its processing in which FRR protection is not possible or practical.

For entry IEAVTRTL: This entry is used by SLIP entry and by RTM when a SLIP function is to be performed. This entry schedules an SVC DUMP or places the system in a wait state via module IEESTPRS.

For entry SLIP2FRR: This is the recovery code for the IEAVTRTL ENTRY function. It frees any acquired resources obtained by IEAVTRTL

For entry RCOVRDMP: This entry defines an FRR that provides recovery for IEAVTRTS when dump options are being copied into the SDWA.

For entry RCOVRCMS: This entry defines an FRR that provides recovery for IEAVTRTS if the CMSET instruction used to enter the FRR's addressing environment fails.

For entry FREESRBS: This entry frees the SRB and EEDs obtained during XMABTERM processing in IEAVTRTM. This routine is a task and address space termination resource manager. It receives control when in XMABTERM processing an SRB is scheduled but before getting dispatched has its related task or address space terminated.

Entry from:

For entry points FREEDCEL and IEAVTRTR: IEAVTRTS using LPSW or BALR.

For entry point RECVRRTM:

IEAVTRTM using BALR.

For entry point FREESRBS: PURGEDQ using BALR.

Exit to:

For entry FREEDCEL: Caller.

For entry RCOVRCRD: IEAVTRTS; or indirectly to IEAVTRTR's entry point RECVRRTM to terminate RTM1's processing of the error.

For entry RCOVRGTF: IEAVTRTS; or indirectly to IEAVTRTR's entry point RECVRRTM to terminate RTM1's processing of the error.

For entry RECVRRTM: IEAVTRTM or IEAVTRTS when RTM1's processing can continue; or to dispatcher IEAV0DS or SRB dispatcher when RTM1's processing of an error is terminated.

For entry IEAVTRTS: Indirectly to IEAVTRTR at entry point RECVRRTM to terminate RTM1's processing of the error.

For entry FREESRBS: PURGEDQ.

For all remaining entry points: IEAVTRTS.

IEAVTRTS — System Recovery Manager

Operation: This module provides the interface and control between the control program and the functional recovery routines defined to recover the control program.

- Interfaces with the software recording facility (IEAVTRER) to record the software errors
- Interfaces with GTF to provide tracing of FRR recovery
- Interfaces with SLIP (IEAVTSLP) to activate any additional serviceability for the error being handled by RTM1

Entry from: IEAVTRTM via BALR.

Exit to: IEAVTRTM.

Called routines: IEAVTRTR.

IEAVTRT1 — RTM1 Entry Point Processor

Operation: This module serves as an extension to the CALLRTM macro instruction and creates a common interface for the mainline RTM1.

Entry point: PROGCK from caller via BALR. At this entry, the module gathers data pertinent to a program interruption for mainline RTM1.

Exit to: IEAVTRTM via BALR.

Entry point: RESTART from caller via BALR. At this entry, the module gathers data pertinent to a restart interruption for mainline RTM1.

Exit to: IEAVTRTM via BALR.

Entry point: SVCERR from caller via BALR. At this entry, the module gathers data pertinent to a program issuing an SVC in a locked, disabled, EUT (enabled, unlocked task that has established an FRR), or SRB environment.

Exit to: IEAVTRTM via BALR.

Entry point: SVCERR from caller via BALR. At this entry, the module saves the caller's registers and establishes a recovery environment.

Exit to: IEAVTRTM via BALR.

Entry point: STERM from caller via BALR. At this entry, the module saves the caller's return address and establishes a recovery environment.

Exit to: IEAVTRTM via BALR.

Entry point: DATERR from caller via BALR. At this entry, the module checks to determine whether the DAT error occurred in the home address space or in another address space. If the error occurred in the home address space, the module gathers data pertinent to a translation failure and passes control to mainline RTM1. Otherwise, it obtains, initializes, and schedules an SRB to run in the master's address space. (See entry point DATMEM in IEAVTRT1.)

Exit to: IEAVTRTM using BALR if the DAT error occurred in the home address space; PROGCK using BALR if the DAT error occurred in another address space.

Entry point: XABTERM from caller via BALR. At this entry, the module saves the caller's registers in a caller-provided save area and establishes a recovery environment.

Exit to: IEAVTRTM via BALR.

Entry point: CABTERM from caller via BALR. At this entry, the module saves the caller's registers in an RTM WSA and establishes a recovery environment.

Exit to: IEAVTRTM via BALR.

Entry point: MEMTERM from caller via BALR. At this entry, the module saves the caller's registers in a caller-provided save area and establishes a recovery environment.

Exit to: IEAVTRTM via BALR.

Entry point: IEAVTRTN via LPSW. This entry is the machine check reentry, at which the module sets up an interface for software recovery from a hardware error.

Exit to: IEAVTRTM via BALR.

Entry point: IEAVTRTX from SRB dispatcher via LPSW. This entry is the cross address space ABTERM reentry, at which the module sets up an interface for the ABTERM function of RTM1.

Exit to: IEAVTRTM via BALR.

Entry point: IEAVTRTZ from IEAVTRTM on return (BR 14). At this entry, the module sets up the final exit linkage back to the mainline supervisor as directed by RTM1.

Exit to:

For retry: Branch via register 15 which is loaded with the retry address returned by RTM1.

For MCH: BR 14.

For restart resume: LPSW of old PSW stored by the restart interrupt handler.

Dispatcher: Branch to the address in CVT0DS.

For SRB: Branch to the address in CVTSRBRT.

For SVC: Branch to the address in CVTEXPRO.

Caller: BR 14.

For EXIT prolog: Branch to the address in CVTEXPRO.

Entry point: DATMEM from the SRB dispatcher via LPSW. This section of code is an SRB scheduled by this module at the DATERR entry point when a DAT error occurs in an address space other than the home address space. This entry point terminates the address space in which the DAT error occurred.

Exit to: SRB dispatcher by means of BR 14.

Called routines: IEAVELKR.

Entry point: DATMEFRR from IEAVTRTS via LPSW. This entry point is the FRR for the DATMEM entry point in IEAVTRT1. DATMEFRR frees the resources (an extended error descriptor block (EED) cell, the DISP lock, and the global intersect) obtained by DATMEM.

Exit to: IEAVTRTS by means of BR 14.

IEAVTRT2 — RTM2 Initialization (SVC 13 Entry Point)

Operation: This module obtains and initializes the work area, and controls routing among major parts of RTM2 (including mainline, exit processing, and critical error handling). It also interfaces with SLIP (IEAVTSLP) to activate any additional error processing for the error being handled by RTM2.

Entry from: SVC FLIH.

Exit to: IEAVTRTE via BALR.

Error exit: Branch to dispatcher via STATUS.

IEAVTRV — Translate Real to Virtual

Operation: This module is given a real address as input. If the real storage page is assigned to a page of virtual space, IEAVTRV returns the corresponding virtual address and ASID.

Entry from: At IEAVTRV from any routine, for 24 bit real address passed as input. At IEAVTRV3 from any routine for 26 bit real address passed as input.

Exit to: Caller.

IEAVTSBP — Task Recovery Resource Manager

Operation: This module transfers the ownership of eligible SCBs to the new RB during XCTL processing. It purges SCBs from the queue when an RB issues an exit or entry to IEAVTSBP for cleanup after an ATTACH failure.

Entry from:

IEAVEAT0 (because of an ATTACH failure) by means of a BALR to the entry point address of IEAVTSBP contained in CVTSCBP.

IEAVEDR (for EXIT processing) by means of a BALR to the entry point of IEAVTSBP contained in CVTSCBP.

IEAVLK00 (for XCTL processing) by means of a BALR to the entry point address of IEAVTSBP contained in CVTSCBP.

IEAVTRTM (for RTM processing) by means of a BALR to the entry point address ESTAECHK contained in a V(con).

Exit to: Caller - with return code 0 or 4.

IEAVTSCB — SCB FREEMAIN Routine

Operation: This module limits the number of cells on the free SCB queue and frees all excess cells.

Entry from: IEAVTSKT via BALR.

Exit to: IEAVTSKT.

Called routines: IEAVGM00.

IEAVTSDC — SVC Dump Clean Up Routine

Operation: This module cleans up SVC dump resources after an SVC DUMP has completed. It frees any locks held and writes an end-of-file mark on the dump data set. IEAVTSDC makes the tasks of the address space being dumped dispatchable, it also makes the system dispatchable (if it is not already from a previous SVC DUMP process). IEAVTSDC issues two messages, IEA911E (signaling dump completion to the operator) and IEA984E (signaling end-of-volume on the tape dump). IEACVPST branches to a secondary entry point, IEAVTSDA, turns off the RTCTDSST (data set full) bit in the RTCT for the new SYS1.DUMP data set tape.

Entry from: IEAVAP00 (synchronous SVC DUMP via BALR), IEAVTSDT (scheduled SVC DUMP via BALR).

Secondary entry point: IEAVTSDA.

Exit to: Caller.

IEAVTSDF — SVC Dump GTF Trace Processor

Operation: If the trace table is a requested part of the SVC DUMP and GTF is active, this module invokes GTF to obtain the saved GTF trace records. It calls IEAVTSDP to write these records on the dump.

Entry from: IEAVAD00 via BALR (synchronous SVC DUMP), IEAVTSDT via BALR (scheduled SVC DUMP).

Exit to: Caller.

IEAVTSDG — SVC Dump Global Storage Processor

Operation: This module adds address ranges to the SVC dump address range table for the following storage areas: nucleus (NUC), PSA, SQA, CSA, and TRT (if supervisor trace is active). It calls IEAVTSD0 to write the storage areas described by the address ranges in the address range table on the dump data set.

Entry from: IEAVAD00 via BALR – synchronous SVC dump. IEAVTSDT via BALR – scheduled SVC dump.

Exit to: Caller.

Modules called: IEAVTSD0 – output processor.

IEAVTSDH — SVC Dump Data Set Initialization

Operation: This module chooses the dump data set for SVC dump. The SDUMP 4K buffer and the AMDPRDMP header record are written on the dump data set. This module uses console interface instructions to obtain the console loop trace buffer. If the console loop trace data has not been previously read, the trace buffer is written on the dump data set. The SVC dump address range table is initialized. Message IEA994A is issued to the operator if all SYS1.DUMP data sets are full and the requested dump cannot be taken. Tape dump data sets are considered full when no tape is mounted.

Entry from: IEAVAD00 via BALR – synchronous SVC DUMP. IEAVTSDT via BALR – scheduled SVC DUMP.

Exit to: Caller.

Modules called: IEAVTSDP (entry point in IEAVTSDO).

IEAVTSDI — SVC Dump Initialization Routine

Operation: This module acquires storage for the SDUMP work area, the SDUMP buffer and the summary dump work area. The SDUMP work area, the summary dump module (IEAVTSSD), and modules IEAVTSDT, IEAVTSDH, IEAVTSD0, and

IEAVTSDW are page fixed in real storage. The dump data set table in the RTM control table (RTCT) is initialized with information about the SYS1.DUMP data sets. This module places the entry point of module IEAVTSDR in the CVT; the entry point of module IEAVTSSD in the summary dump work area (SMWK); and the address of the attention routine, IEAVTSDA, in attention table index 32, maintained by IOS.

Entry from: IEAVNPA6 via BALR.

Exit to: Caller.

IEAVTSDL — SVC Dump Local Storage Processor

Operation: This module adds address ranges to the SVC dump address range table for the following storage areas: LSQA, SWA, RGN (including LSQA and SWA), and LPA. Module IEAVTSDO is called to write the storage areas described by the address ranges in the address range table on the dump data set.

Entry from: IEAVAD00 via BALR – synchronous SVC dump. IEAVTSDT via BALR – scheduled SVC dump.

Exit to: Caller.

Modules called: IEAVTSDO – output processor.

IEAVTSDO — SVC Dump Output Processor

Operation: This module performs I/O to write on the dump data set the storage areas described in the SVC dump address range table. Secondary entry point IEAVTSDP is used to write one record on the dump data set. Secondary entry point IEAVTSEO allows SVC dump exits to write data on the dump data set.

Entry from: IEAVTSDG via BALR – write out global storage. IEAVTSDL via BALR – write out local storage.

Secondary entry points: IEAVTSDP and IEAVTSEO.

Exit to: Caller.

IEAVTSDR — SVC DUMP Resources Manager

Operation: This module will clean up SVC dump resources if an address space fails in which SVC dump is active.

Entry from: IEAVTMMT via BALR.

Exit to: IEAVTMMT.

IEAVTSDT — SVC Dump Task (one in every address space) attached at address space creation time

Operation: This module performs SVC dump processing in each address space that is the target of a scheduled SVC dump request. It is awakened in an address space when the SRB routine, scheduled by IEAVTSDX, executes the POST function. It calls the SVC dump service modules IEAVTSDH, IEAVTSDG, IEAVTSDL, IEAVTSDF and IEAVTSDC to dump the requested storage of the address space in which this routine is awakened.

Entry from: Attached by IEAVTSDX (via POST) during address space creation, at DUMPREQ for each dump request of the address space in which this module is resident. DUMPREQ is a label in IEAVTSDT where processing starts when the task is resumed.

Exit to: None. This is a never ending task (as long as the address space exists).

IEAVTSDU — SVC Dump Exit Interface Routine

Operation: This module allows component dump exit routines to be invoked by SVC dump in order to add component-related data to the dump. A general parameter list is passed to the exit routine. The exit routines to be called are defined in CSECT IEAVTSXT in SYS1.LINKLIB. The exit definitions give the name and attributes of each exit routine.

Entry from: IEAVAD00 and IEAVTSDT via BALR.

Exits called: IEEMB879, ISGDSMP.

Exit to: Caller.

IEAVTSDW — SVC Dump Summary Dump Writer

Operation: This module puts the data in the summary dump buffers into AMPRDMP format and invokes module IEAVTSDP to write these constructed records on the dump data set. In cases where the real storage buffer frame address is greater than 16 Mb or the real address of the SVC dump output buffer is greater than 16 Mb, IEAVTSDN calls IEAVWNDE to obtain virtual addressability to both frames. As the contents of each frame is written on the dump data set, this module returns the frame to RSM.

Entry from: IEAVTSDT via BALR.

Called routines: IEAVTSDP, IEAVWNDE, IEAVTSV0.

Exit to: Caller.

IEAVTSDX — SVC Dump Branch Entry Module

Operation: This module schedules an SVC dump in the address space(s) specified by the caller. It schedules an SRB in the requested address space(s). The SRB routine posts the dump task (IEAVTSDT) in the address space(s) requested.

Entry from: Caller via BALR.

Exit to: Caller.

Called routines: IEAVSETS, IEAVSY50, IEAVTSSD, IEAVTSSV, IRARMINT.

IEAVTSFR — SETFRR

Operation: The SETFRR macro calls this module to update the FRR stacks. Depending on the request, IEAVTSFR adds or deletes a current stack entry, purges all the entries from the current stack, flushes all the entries from the normal stack, or flushes all the entries from the normal stack and then adds an entry to this stack.

Entry from: SETFRR macro.

Exit to: Caller.

IEAVTSIG — SLIP PER RISGNL Routine

Operation: This module adjusts the PSW PER indicators in the I/O, external, and SVC new PSWs, and loads the PER control registers (9-11) with appropriate values when PER monitoring is being activated or deactivated.

Entry from: IEAVTGLB via BALR.

Exit to: Caller.

IEAVTSIN — FRR Stack Initialization

Operation: IEAVTSIN is an executable macro included by IEAVNIPO and IEEVCPU to perform FRR stack initialization for a given processor.

Entry from: This inline expansion is included by IEAVNIP0 (during system initialization) and by IEEVCPO (during vary processor processing).

Exit to: Caller.

IEAVTSKT — Task Termination

Operation: This module handles the releasing of a terminating task resource.

Entry from: IEAVTRTE via BALR.

Exit to: IEAVTRTE.

IEAVTSLB — SLIP Action Processor — Part 2

Operation: This module routes control to the SLIP trap match routines ((IEAVTSL1 or IEAVTSL2) depending on the parameters found in the SCVA)) and processes the MATCHLIM and PRCNTLIM keywords.

Entry from: IEAVTSLP.

Exit to: IEAVTSLE.

Called routines: IEAVTSL1, IEAVTSL2.

IEAVTSLE — SLIP Action Processor — Part 3

Operation: This module initiates message processing requested by IEAVTSLP when a trap has been automatically disabled. IEAVTSLE also removes the recovery environment established by IEAVTSLP, and returns to the caller of IEAVTSLP.

Entry from: IEAVTSLB, IEAVTSLR.

Exit to: The caller of IEAVTSLP.

IEAVTSLP — SLIP Action Processor — Part 1

Operation: This module receives control when SLIP is active from either an RTM routine as part of normal processing, or from IEAVTPER after a PER interruption. IEAVTSLP establishes error recovery to match the environment of the caller, serializes the SCE chain, and passes control to IEAVTSLB to continue SLIP action processing.

Entry from: IEAVTRTS, IEAVTRT2, IEAVTRTM, or IEAVTPER.

Exit to: IEAVTSLB.

IEAVTSLR — SLIP Processor Recovery

Operation: This module provides the FRR recovery for the SLIP processor modules (IEAVTSLP, IEAVTSLB, IEAVTSLE, IEAVTSL1, IEAVTSL2, IEAVTSLS) and the address conversion routine (IEAVTADR) when it is called by SLIP. If a recoverable error has occurred, this module attempts a retry at an appropriate point in the SLIP processor module. If the error is not recoverable, IEAVTSLR requests percolation.

Entry from: IEAVTRTS (FRR router).

Exit to: IEAVTRTS.

IEAVTSLS — SLIP Processor Service Routine

Operation: This module is called to: determine the system mode, issue a DUMP, or schedule IEAVTGLB to turn off PER monitoring.

Entry from: IEAVTSLB, IEAVTSL1, IEAVTSL2.

Exit to: Caller.

IEAVTSL1 — SLIP Trap Matching Routine — Part 1

Operation: This module determines whether the event qualifier keywords specified on the SLIP command match the current system conditions. IEAVTSL1 processes the ADDRESS, ASID, COMP, ERRTYP, JOBNAME, JSPGM, LPAMOD, MODE, RANGE, and ASIDSA event qualifier keywords.

Exit to: Caller.

IEAVTSL2 — SLIP Trap Matching Routine — Part 2

Operation: This module: determines whether the event qualifier keywords specified on the SLIP command match the current system conditions (IEAVTSL2 processes the DATA and PVTMOD event qualifier keywords). When a match is found for all event qualifier keywords, IEAVTSL2 is called to perform the action specified in this trap.

Entry from: IEAVTSLB.

Exit to: Caller.

IEAVTSR1 — ITERM Processor

Operation: This module processes CALLRTM macros that specify TYPE=ITERM and COMPCOD=cc. RTM uses IEAVTSR1 to abnormally terminate an interrupted SRB or TCB. IEAVTSR1 prepares this interrupted SRB or TCB for the abend.

Entry from: Issuer of CALLRTM using BALR.

Exit to: Caller.

Entry point SR1RETRY: This entry point prepares IEAVTSR1 for an abend by placing the SVC 13 address in the old PSW. It also deletes the recovery environment.

Entry from: SR1FRR.

Exit to: Caller.

Entry point SR1FRR: This entry point is the FRR for IEAVTSR1. It puts diagnostic information into the SDWA. If the error occurred in the GETCELL routine, it retries. Otherwise, it frees the EED and requests recording and percolation.

Entry from: IEAVTRTS via LPSW.

Exit to: IEAVTRTS by means of BALR.

IEAVTSSD — SVC DUMP Disabled Summary Dump Processor

Operation: For branch entries to SVC dump, this module invokes RSM (IEAVPRSB) and obtains a real storage buffer from the the real storage backing the pageable link pack area. It calls IEAVTSSM to move volatile system information into these real storage frames. Module IEAVTSDW later writes the saved information on the dump data set in AMDPRDMP format.

Entry from: IEAVTSDX via BALR.

Exit to: Caller.

IEAVTSSE — SVC Dump Enabled Summary Dump Processor

Operation: For nonbranch entries to SVC DUMP, this module writes a summary dump on the dump data set. This summary dump includes the following storage areas:

- Storage areas specified with the SUMLIST keyword on the SDUMP macro
- Supervisor trace table

- RTM2WA (if one exists)
- 2K of storage plus and minus unique addresses found in the registers at the time of error.

Entry from: IEAVTSDT via BALR, IEAVAD00 via BALR.

Exit to: Caller.

IEAVTSSH — SLIP Space Switch Handler

Operation: When a unit of work changes the address space in which it is executing, this module adjusts the PER bit in the PSW according to the criteria established on the SLIP command.

Entry from:

For entry point IEAVTSSH: IEAVTPER when a space switch interrupt occurs.

For entry point IEAVTSS1: Any issuer of the CMSET SET macro when there is an active SLIP trap for that macro.

For entry point IEAVTSS2: Any issuer of the CMSET RESET, CHKAUTH=YES macro when there is an active SLIP trap for that macro.

For entry point IEAVTSS3: Any issuer of the CMSET RESET, CHKAUTH=NO macro when there is an active SLIP trap for that macro.

Exit to:

For entry point IEAVTSSH: IEAVTPER using a BR 14.

For entry points IEAVTSS1, IEAVTSS2, IEAVTSS3: Caller using a BR 14.

IEAVTSSM — SVC Dump Summary Dump Data Move Routine

Operation: For disabled and suspend summary dumps, this module moves volatile system data to the real storage buffer or the virtual storage buffer located in the DUMPSRV address space. In cases where the real storage buffer frame address is greater than 16 Mb or the real address of the SVC dump output buffer is greater than 16 Mb, IEAVTSSM calls IEAVWNDE to obtain virtual address ability to both frames.

Entry from: IEAVTSSD using BALR. IEAVTSSV using BALR.

Called routines: IEAVWNDE.

Exit to: Caller.

IEAVTSSV — SVC Dump Suspend Summary Dump Processor

Operation: For branch entries to a SVC dump with the SUSPEND=YES keyword specified, this module calls IEAVTSSM to move volatile system information to the virtual storage buffer located in the DUMPSRV address space.

Entry from: IEAVTSPX (SCHSRB) using BALR.

Exit to: Caller.

IEAVTSSX — Space Switch Extension

Operation: This module adjusts the PER bit in the PSW for RTM1. IEAVTSSX performs the same function as IEAVTSSH, but is tailored to meet the needs of RTM1.

Entry from: IEAVTRTS (RTM1) using a BALR.

Exit to: IEAVTRTS using a BR 14.

IEAVVCRA — Communication Task Console Attention Processor

Operation: The console attention processor receives control as a result of someone pressing the console's attention key. Pressing the attention key causes an attention interrupt. As a result of the attention interruption, this routine posts the communication task.

Entry from: IEACVPST.

Exit to: The POST status routine of the IOS handler via a BR 14 instruction.

IEAVVCRX — External Interrupt Key Processor

Operation: This module receives control from the external first level interrupt handler to post the communication task.

Entry from: IEAVEEXT.

Exit to: The external first level interrupt handler by branching to the address in register 2.

IEAVVCTR — SVC 72 Router

Operation: IEAVVCTR determines the appropriate device support processor (DSP) to process the communications task service.

Entry from: Issuer of SVC 72.

Exit to:

- IEAV1052 for a 1052 printer-keyboard, 3210 console printer-keyboard, 3215 console printer-keyboard, or 3213 console printer supported as a console.
- IEAV1443 for a 1443 printer, 1403 printer, or 3211 printer supported as a console.
- IEAV2540 for a 2540 card reader punch, 2501 card reader, 2520 card reader punch, 3505 card reader, or 3525 card punch supported as a console.
- IEEC2740 for a 2740 communications terminal supported as a console.
- IEECVETW for a 3284/3286 printer supported as a console.
- IEECVET1 for a graphics device supported as a console.

IEAVVRP2 — Reply Processor - Stage 2

Operation: This module checks whether the text of a user's buffer area is valid. If the buffer is valid, IEAVVRP2 moves the reply text into the buffer.

Entry from: IEECB811 (via an SRB).

Exit to: Dispatcher.

IEAVVWTO — Communication Task Single-Line WTO and WTOR Service Routine (SVC 35)

Operation: The WTO and WTOR service routine processes the SVC 35 issued by the WTO and WTOR macro expansion. It also receives control for the multiple-line WTO (MLWTO) and write-to-programmer (WTP) macros. For these two, IEAVVWTO branches to the MLWTO (IEAVMWTO) and WTP (IGC0203E) routines, respectively. When a WTO or WTOR message is sent to a console, IEAVVWTO posts the communication task.

Entry from: IEAVESVC.

Exit to: The SVC first level interrupt handler by branching to the address in register 14.

Error exit: When a WTO message has a length of zero bytes, the service routine branches to the SVC first level interrupt handler at the address in register 14. For all other errors, IEAVVWTO gives control to RTM via an ABEND macro. RTM terminates the issuer of the WTO or WTOR macro with completion code X'D23'.

Called routines: IEAVH600, IEAVMWTO, IEAVM700, IEECVCTE, IGC0203E (CSECT in IEEJB840), IEFJSREQ.

IEAVWND — RSM Window Service Routine

Operation: This module provides a means of making one or two frames above the 16 megabyte (Mb) line temporarily virtually addressable.

Entry from:

For entry point IEAVWNDI: IEAVGFAX. This entry point does not set an FRR. The FRR is already in place because it was set by the calling RSM routines.

For entry point IEAVWNDE: IECVDERP, IEAVTSSD, IEAVTSSM, IEAVTSDW, and IEEVALST. This entry point always sets an FRR.

Exit to: Caller.

Error exit:

For entry point IEAVWNDE: For input parameter errors, this module issues an ABEND macro. For other errors, IEAVWND passes control to IEAVRCV. IEAVRCV retries to entry point WNDRETRY (within IEAVWND). WNDRETRY cleans up and returns to the caller of IEAVWND with a return code of 8.

For entry point IEAVWNDI: IEAVRCV receives control to record information about the error and then percolates.

IEAVXDOM — Communication Task DOM Service Routine (SVC 87)

Operation: The DOM service routine processes the SVC 87 that is issued by the DOM macro expansion. This routine builds a DOM control block (DOMC) with the

message IDs of the messages to be deleted, then posts the communication task.

Entry from: IEAVESVC.

Exit to: The SVC first level interrupt handler by branching to the address in register 14.

Error exit: RTM via an ABEND macro. RTM terminates the user of the DOM macro with completion code X'157'.

Called routines: IEAVH600.

IEAVXEC0 — PC/AUTH Service Routine that Connects an Entry Table

Operation: This module connects entry tables to specified indexes in the linkage table of the home address space.

Entry from:

For entry point IEAVXEC0: Issuer of an ETCON macro.

For entry point IEAVXC05: PC/AUTH service routines' FRR (IEAVXPCR).

Exit to: Caller by issuing a PT instruction.

Error exit: RTM via an ABEND macro.

Called routines: IEAVXSET at entry point IEAVXACK, and IEAVEBBR.

IEAVXECR — PC/AUTH Service Routine that Creates an Entry Table

Operation: This module builds an entry table and its associated entry table information block (ETIB) for the home address space using an entry table description list provided by the caller.

Entry from:

For entry point IEAVXECR: Issuer of an ETCRE macro.

For entry point IEAVXC03: PC/AUTH service routines' FRR (IEAVXPCR).

Exit to: Caller by issuing a PT instruction.

Error exit: RTM via an ABEND macro.

Called routines: IEAVQCDSR (CDE search routine), IEAVVMSR (LPDE search routine), and IEAVXEPM (nucleus entry point search routine).

IEAVXEDE — PC/AUTH Service Routine that Deletes an Entry Table

Operation: This module deletes a specified entry table and its associated entry table information block (ETIB). If the caller requests the purge option, IEAVXEDE also disconnects the entry table from all linkage tables.

Entry from: Issuer of an ETDES macro.

Exit to: Caller by issuing a PT instruction.

Error exit: RTM via an ABEND macro.

IEAVXEDI — PC/AUTH Service Routine that Disconnects an Entry Table

Operation: This module disconnects an entry table(s) from the linkage table of the caller's home address space.

Entry from: Issuer of an ETDIS macro.

Exit to: Caller by issuing a PT instruction.

Error exit: RTM via an ABEND macro.

IEAVXEPM — Program Call Nucleus Entry Point Search Module

Operation: When called at entry point IEAVXEPM, this module searches the nucleus entry point table (IEAVXNET) for a specified entry point, and returns the entry point address to the caller. IEAVXEPM also receives control at entry point IEAVXABE when a program call (PC) instruction references a default entry table entry. It issues an X'0D6' abend.

Entry from:

For entry point IEAVXEPM: IEAVXMAS during PC/AUTH address space initialization (see *System Initialization Logic*), and IEAVXECR when creating an entry table.

For entry point IEAVXABE: Branched to from a PC instruction.

Exit to:

For entry point IEAVXEPM: Caller.

For entry point IEAVXABE: RTM via an ABEND macro.

Error exit: RTM via an ABEND macro.

IEAVXLFR — PC/AUTH Service Routine that Frees Linkage Indexes

Operation: This module frees the linkage indexes specified by the caller.

Entry from:

For entry point IEAVXLFR: Issuer of an LXFRE macro.

For entry point IEAVXC02: PC/AUTH service routines' FRR (IEAVXPCR).

Exit to: Caller by issuing a PT instruction.

Error exit: RTM via an ABEND macro.

Called routines: IEAVXEDI via an ETDIS macro.

IEAVXLRE — PC/AUTH Service Routine that Reserves a Linkage Index

Operation: This module reserves linkage indexes for the caller's home address space.

Entry from:

For entry point IEAVXLRE: Issuer of the LXRES macro.

For entry point IEAVXC01: PC/AUTH service routines' FRR (IEAVXPCR).

Exit to: Caller by issuing a PT instruction.

Error exit: RTM via an ABEND macro.

IEAVXMAS — See OS/VS2 System Initialization Logic for a module description of IEAVXMAS.

IEAVXMIN — Memory Initialization Routine for PC/AUTH Services

Operation: During address space initialization, IEAVEMIN gives IEAVXMIN control in SRB mode. IEAVXMIN connects the address space being initialized to the system authorization table (SAT) and system linkage table (SLT). It also sets the address space's authorization index (AX) to zero and indicates that the address space can be accessed in cross memory mode.

Entry from:

For entry point IEAVXMIN: IEAVEMIN.
For entry point XMINFRR: RTM.

Exit to: Caller.

IEAVXNEP — Nucleus Entry Point Table Module

Operation: This module contains a table (IEAVXNET) that includes:

- The names of routines in the nucleus that can be entered using a program call (PC) instruction
- A VCON for each routine's entry point address

Reference from: IEAVXMAS (described in *System Initialization Logic*) and IEAVXECR.

Exit to: Not applicable (This module does not contain executable code.)

IEAVXPAM — PC/AUTH Resource Manager

Operation: When an address space or a task that owns a cross memory resource is being terminated, this module receives control to recover the PC/AUTH resources. It:

- Resets the authorization index, authorization table origin, and the linkage table designator of the terminating address space or task.
- Disconnects the address space from any cross memory services available to it, except those available via the system linkage table (SLT).
- Frees the PC/AUTH resources owned by the terminating address space or task.

Entry from:

For entry point IEAVXPAM: IEAVTSKT via a PC instruction issued by the PCARM macro IEAVTMMT.

For entry point IEAVXC0C (cleanup exit): PC/AUTH service routines' FRR (IEAVXPCR).

Exit to: Caller by issuing a PT instruction.

Error exit: RTM via an ABEND macro.

Called routines: IEAVEBBR.

IEAVXPCR — PC/AUTH Service Routines' FRR

Operation: This module provides recovery for the program call/authorization (PC/AUTH) service routines and for itself.

Entry from: IEAVTRTS (RTM) by issuing an LPSW instruction.

Exit to: RTM by issuing a BR 14 instruction.

Error exit: RTM by issuing a BALR instruction.

Called routines: IEAVEQV1 (single-threaded queue verification routine), IEAVEQV3 (double-threaded queue verification routine), IEAVXC01 (cleanup exit for IEAVXLRE), IEAVXC02 (cleanup exit for IEAVXLFR), IEAVXC03 (cleanup exit for IEAVXECR), IEAVXC05 (cleanup exit for IEAVXEC0), IEAVXC07 (cleanup exit for IEAVXARE, IEAVXC0A (cleanup exit for IEAVXAXS, IEAVXC0B (cleanup exit for IEAVXATS, IEAVXC0C (cleanup exit for IEAVXPAM).

IEAVXRFE — PC/AUTH Service Routines that Reserve, Free, or Extract an Authorization Index

Operation: This module performs three cross memory authorization functions. It:

- Reserves one or more authorization indexes (AXs) for the home address space (entry point IEAVXARE)
- Frees one or more AXs (entry point IEAVXAFR)
- Extracts the AX value of a specified address space and returns it to the caller (entry point IEAVXAEX)

Entry from:

For entry point IEAVXARE: Issuer of an AXRES macro via PC.

For entry point IEAVXAFR: Issuer of an AXFRE macro via PC.

For entry point IEAVXAEX: Issuer of an AXEXT macro via PC.

For entry point IEAVXC07 (cleanup exit): PC/AUTH service routines' FRR (IEAVXPCR).

For entry point IEAVXC08 (cleanup exit): PC/AUTH service routines' FRR (IEAVXPCR).

Exit to: Caller by issuing a PT instruction.

Error exit: RTM via an ABEND macro.

IEAVXSEM — See OS/VS2 System Initialization Logic for a module description of IEAVXSEM.

IEAVXSET — PC/AUTH Service Routines that Set an Authorization Table Entry or an Authorization Index

Operation: This module performs four functions:

- Sets an authorization table (AT) entry for the caller's home address space (entry point IEAVXATS)
- Sets an authorization index (AX) for the caller's home address space (entry point IEAVXAXS)
- Determines whether a specified address space is authorized to issue PT and SSAR instructions to another specified address space (entry point IEAVXACK)
- Determines whether a specified address space has an AX value of 1 (entry point IEAVXACK)

Entry from:

For entry point IEAVXATS: Issuer of an ATSET macro.

For entry point IEAVXAXS: Issuer of an AXSET macro.

For entry point IEAVXACK: IEAVXEC0 via a BALR.

For entry point IEAVXC0A (cleanup exit): PC/AUTH service routines' FRR (IEAVXPCR).

For entry point IEAVXC0R (cleanup exit): PC/AUTH service routines' FRR (IEAVXPCR).

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEAVEBBR.

IEAVXSFM — System Function Table Module

Operation: This module contains the system function table (SFT), which contains pre-allocated PC numbers for system-wide program call services.

Entry from: Not applicable. (This module contains no executable code.)

Exit to: Not applicable. (This module contains no executable code.)

IEAVXSTK — PCLINK STACK, UNSTACK, and EXTRACT Service Routine

Operation: This module performs the functions requested by PCLINK macros. It:

- Creates a PCLINK stack element in which to save cross memory linkage information when entered via a PCLINK STACK macro
- Removes element(s) from the PCLINK stack when entered via a PCLINK UNSTACK macro)
- Returns information from a specified PCLINK element when entered via a PCLINK EXTRACT macro

Entry from:

For entry point IEAVXSTS: Issuer of a PCLINK STACK SAVE=YES macro.

For entry point IEAVXSTN: Issuer of a PCLINK STACK SAVE=NO macro.

For entry point IEAVXUNS: Issuer of a PCLINK UNSTACK SAVE=YES macro.

For entry point IEAVXUNN: Issuer of a PCLINK UNSTACK SAVE=NO macro.

For entry point IEAVXEXT: Issuer of a PCLINK EXTRACT macro.

For entry point FRRFRSTK (FRR for PCLINK STACK processing: RTM.

For entry point FRRUNSTK (FRR for PCLINK UNSTACK processing): RTM.

For entry point FRRUNSK2 (FRR for FRRUNSTK): RTM.

For entry point RETRYUNS (retry routine for FRRUNSTK): RTM.

Exit to: Caller.

Error exit: RTM via an ABEND macro, or if the macro issuer specified an ERRET, to the ERRET.

IECDAFMT — Data Management Control Block Formatter Mainline

Operation: This module locates the data management control blocks associated with the TCB passed in the parameter list and determines if they are to be formatted. If they are, it calls the appropriate control block formatter for each data management control block.

Entry from: IEAVAD01 or AMDPRUIM.

Exit to: Caller.

IECDAFT1 — Data Management Control Block Format Module

Operation: This module formats either the DEB, DCB, or IOB passed in the parameter list depending on which entry is called. The DCB formatter determines the access method type and formats the DCB accordingly. It also returns this information to the caller for use in locating the IOBs. The IOB formatter also uses the access method information to format the IOB (ICB,LCB). All entry points send a return code to the caller indicating success or failure in formatting. In addition, the DCB formatter sets a return code indicating failure if the access method does not have IOBs to be formatted.

Entry from: IECDAFMT at entry points IECDADCB, IECDADEB, IECDAIOB, and IECDAFT1.

Exit to: Caller.

IECIOFMT — IOS Control Block Formatter Mainline

Operation: This module locates the IOS control blocks associated with the TCB passed in the parameter list and determines if they are to be formatted. If they are,

it calls the appropriate control block formatters for each IOS control block.

Entry from: IEAVAD01 or AMDPRUIM.

Exit to: Caller.

IECIOFT1 — IOS Control Block Format Module

Operation: This module formats the EXCPD(XDBA) passed in the parameter list or the UCBs passed in the parameter list. For UCBs it determines the UCB type and formats it accordingly. It sends a return code to the caller indicating success or failure in formatting.

Entry from: IECIOFMT at entry points IECIOEXD and IECIOUCB.

Exit to: Caller.

IEDAY3 — LOGON Synchronization Module

Operation: This routine synchronizes the LOGON process for TIOC by preventing memory creation from proceeding until the TIOC LOGON routine has fully initialized the ASCB.

Entry from: IEEVWAIT.

Exit to: IEEVEMCR.

IEEAB400 — WTP Buffer Routine

Operation: This module puts WTP messages into buffers and, if the messages are also WTO messages, issues the WTO macro. When the buffer becomes full, this module calls IEEAB401 to issue the WTPs.

Entry from: IEFAB4FD, IEFAB4DD, IEFAB4E4.

Exit to: Caller.

Called routines: IEEAB401.

IEEAB401 — WTP PUT Routine

Operation: This module issues the PUT macro to write any WTP messages present in the buffer built by IEEAB400.

Entry from: IEEAB400, IEFAB4A0, IEFAB4DD, IEFAB421, IEFAB4E4, IEFAB4E8, IEFBB401, IEFBB410.

Exit to: Caller.

Called routines: None.

IEECB800 — DISPLAY/TRACK Command Common Processor

Operation: This module builds the display for a DISPLAY A or TRACK A command. The display consists of system status information about active tasks, jobs, and time sharing terminals. TRACK command requests appear on a graphics (screen) device. DISPLAY command requests appear on either a graphics or a paper device.

Entry from: IEEVWAIT via ATTACH.

Exit to: End of task.

Called routines: IEECB801.

IEECB801 — Issuer of WTO and TPUT for DISPLAY and TRACK Commands

Operation: This module issues the multi-line WTO (MLWTO) or iterative TPUTs for the DISPLAY and TRACK commands.

Entry from: IEECB800, IEECB804, IEECB808.

Exit to: Caller.

IEECB804 — DISPLAY R Command Processor

Operation: This module displays on an operator's console, the text for the following types of messages:

- Messages awaiting replies (WTORs)
- Immediate action messages
- Eventual action messages

Addresses of devices that need to be made ready or need operator intervention are also displayed.

Entry from: Attached by IEEVWAIT.

Exit to: Caller.

IEECB805 — SET MPF Processor

Operation: This module builds a sorted table of message IDs and message prefixes for messages that are to be suppressed. It initializes the table with entries from the MPFLSTxx SYS1.PARMLIB member, where xx is the operand specified on the SET MPF command.

Entry from: IEEMB811.

Exit to: IEEMB811.

Called routines: IEEMB878, IEEMB882, IEE0503D, IEFJSIMW.

IEECB806 — TRACE Command (MT-ON, MT-OFF, STATUS operands) Processor

Operation: This module syntax checks the command, passes control to IEEMB809 (if appropriate) via the IEETRACE macro, and issues a STATUS message.

Entry from: IEEVWAIT via ATTACH.

Exit to: End of task.

IEECB807 — DISPLAY MPF Processor

Operation: This module displays the entries in the current MPF table built by IEECB805.

Entry from: IEECB808.

Exit to: IEECB808.

Called routines: IEECB808, IEE0503D.

IEECB808 — Command Routine

Operation: This module routes control to the DISPLAY MPF command processor (IEECB807), the VARY global resource serialization command processor (IEECB921), or the DISPLAY global resource serialization command processor (IEECB922). It also provides the message service subroutine (MSGSERV), a secondary entry point in IEECB808, which the command processors call to:

- Build a single line of a message
- Issue a complete message
- Free any remaining write parameter lists (WPLs)

Entry from: IEEVWAIT.

Exit to: IEEVWAIT.

Called routines: IEECB860, IEECB801.

IEECB809 — VARY Device Service Routine

Operation: This module handles all ASSIGN/UNASSIGN and dynamic pathing processing for the MVS VARY Device command.

Entry from: IEECB862, IEE3103D.

Exit to: IEECB862, IEE310D.

Called routines: IECVDPTH, IEEMB814, IEFAUSRV.

IEECB811 — Reply Syntax Checker

Operation: This module validates the command syntax and ID and schedules an SRB for stage 2 processing.

Entry from: IEECB808.

Exit to: IEECB808.

IEECB860 — Command ESTAE Creation/Exit Routine

Operation: This module creates an ESTAE environment for the module that invokes it. If the ABEND STAE interface routine (ASIR) invokes the module, it provides a dump and messages.

Entry from: ABEND STAE interface routine, IEECB808, IEECB913.

Exit to: ABEND STAE interface routine.

IEECB862 — VARY Device ESTAE Exit Routine

Operation: This module is the ESTAE exit routine for VARY Online/Offline/Console processing. It updates the SDWA and takes an SVC dump. Then, if any device was assigned or had dynamic pathing done, but is not online, the device is unassigned and the path group is disbanded. IEECB862 then returns to RTM to percolate to IEECB860.

Entry from: RTM.

Exit to: RTM.

Called routines: IEECB809.

IEECB866 — Console Dump

Operation: This module causes a dump of virtual storage to the current dump data set SYS1.DUMP.

Entry from: IEEVWAIT.

Exit to: System through use of SVC 3.

IEECB900 — VARY CN Syntax Processor

Operation: This module syntax checks the command and passes control to the authority processor for valid commands.

Entry from: IEEVWAIT.

Exit to: End of task.

Called routines: IEECB901.

IEECB901 — VARY CN Console Processor

Operation: This module modifies a console authority value for eligible consoles.

Entry from: IEECB900.

Exit to: Caller.

IEECB904 — VARY Range Processor

Operation: This module creates a VARY command from a VARY device address-range command.

Entry from: IEE3603D, IEE3103D, IEE3303D, IEE4203D, IEE4603D.

Exit to: Caller.

IEECB905 — SLIP Command Processor

Operation: This module sets, modifies, and deletes SLIP traps. It also schedules IEAVTGLB to activate or deactivate PER monitoring.

Entry from: IEEVWAIT via ATTACH.

Exit to: SVC 3 via a BR 14 instruction.

IEECB906 — SLIP ESTAE Routine

Operation: This module provides ESTAE recovery for the SLIP command (IEECB905 and IEECB909), the DISPLAY SLIP command (IEECB907) and the SLIP message module (IEECB908). It cleans up resources and takes an SVC dump. It does not retry.

Entry from: ABEND STAE interface routine.

Exit to: ABEND STAE interface routine.

IEECB907 — Display SLIP Processor

Operation: This module displays SLIP traps either in summary or in detail in response to the DISPLAY SLIP operator command. If the SLIP command processor requests it, IEECB907 also provides detailed trap information.

Entry from:

- IEEVWAIT via ATTACH for DISPLAY SLIP
- IEECB905 via BALR to show options

Exit to: End of task if entered from IEEVWAIT, IEECB905 if entered from IEECB905.

Called routines:

- IEECB860 via BALR and ESTAE exit
- IEECB906 via ESTAE exit
- IEECB908 via BALR
- IEE0503D via BALR

IEECB908 — SLIP Message Module

Operation: This module contains SLIP messages. It formats and writes out the message requested and if the message demands a response, waits for a reply. It also posts IEECB914 to converse with a TSO SLIP user.

Entry from: IEECB905 and IEECB907.

Exit to: Caller.

IEECB909 — SLIP Command Interpreter

Operation: This module scans and interprets all SLIP command input text.

Entry from: IEECB905 via BALR.

Exit to: Caller.

IEECB910 — Display Dump Processor

Operation: This module produces the IEE856I and IEE857I message output in response to the operator command DISPLAY DUMP.

Entry from: IEEVWAIT.

Exit to: Caller.

IEECB911 — Display Dump Message CSECT

Operation: This CSECT contains the message text that IEECB910 uses to construct the lines for its output messages.

Used by: IEECB910.

IEECB912 — Display Dump MLWTO Interface

Operation: This subroutine converts lines of output into MLWTO format.

Entry from: IEECB910.

Exit to: Caller.

IEECB913 — SETSMF Processor

Operation: This module changes one or more individual SMF options as requested in the SETSMF command. In addition, it issues an SSI call to the subsystems whose options have been modified.

Entry from: IEEVWAIT via an ATTACH.

Exit to: IEEVWAIT.

Called routines: IEECB860, IEEMB821, IEEMB823, IEEBM824, IEEMB829, IEFJSREQ via a BALR.

IEECB914 — SLIP TSO Communication Routine

Operation: This module issues a TPUT and TGET to the SLIP TSO user.

Entry from: IKJEFF00 via ATTACH.

Exit to: SVC 3 via BR14.

IEECB915 — SLIP Processor POST Routine

Operation: When it is scheduled by the SLIP processor (IEAVTSLP), this module posts the SLIP command processor (IEECB905). The posting is done when the SLIP processor requests message processing.

Entry from: SRB dispatcher.

Exit to: Caller.

IEECB916 — Display SMF Processor

Operation: This module lists SMF data set status and the SMF options currently in effect.

Entry from: Attached by IEEVWAIT (master scheduler).

Exit to: Caller.

Called routines: IEEMB824, IEEMB833, IEECB860 via BALR.

IEECLEAN — Vary CPU/Channel Cleanup and Recovery Routine

Operation: For the Vary CPU/channel function, this routine issues messages and frees the system resources including storage for recorder blocks and SMF records.

Entry from: IEEVCPR.

Exit to: Caller.

IEECVETA — CONTROL Command Syntax Checker (DIDOCS)

Operation: Checks the syntax of the CONTROL command.

Entry from: IEECVET1, IEECVET4.

Exit to:

- IEECVETK if roll mode or RTME (time interval specification) changes
- IEECVETD to issue warning message
- IEECVETH/P/R/U (device-dependent I/O modules) to rewrite the entry area
- IEECVETD to issue an error message

IEECVETC — Asynchronous Error Processing Module (DIDOCS)

Operation: This module initializes the DCM for a reopen. It processes asynchronous errors and prepares for a console switch after a permanent synchronous error.

Entry from:

- IEECVET1 for asynchronous error processing
- IEECVFTG for asynchronous error processing
- IEECVET4 for asynchronous error processing

Exit to:

- IEECVETE to issue error message
- IEECVFTG to assist in error processing
- IEECVETK to set timer interval
- IEECVETH/P/R/U (device-dependent I/O modules) to write the screen image
- IEAVSWCH to perform console switch
- IEECVET4 to simulate pressing the CLEAR key if the device supports the erase-write-alternate (EWA) command

IEECVETD — Message Module (DIDOCS)

Operation: For full capability consoles, places error or informational messages in the warning line or the instruction line; for message stream or status display consoles, places messages in the warning line.

Entry from: Any DIDOCS module detecting a message-output condition.

Exit to:

- IEECVETH/P/R/U (device-dependent I/O modules) to write the screen.
- IEECVET1 if hold mode is in effect.
- IEECVET3 if roll mode is in effect and an unviewable message is needed.

IEECVETE — Message Module (DIDOCS)

Operation: Places messages into the instruction or warning line and places asynchronous error messages into the entry area.

Entry from:

- IEECVETC to issue asynchronous error message
- IEECVET4 to issue error message
- IEECVETK to issue error message
- IEECVETF to issue error message
- IEECVET6 to issue error message
- IEECVET8 to issue error message

Exit to: IEECVETH/P/R/U (device-dependent I/O modules) to write messages to the screen.

IEECVETF — Light-Pen and Cursor Detect Analyzer (DIDOCS)

Operation: Determines from the location of a light-pen or cursor detect what function is being requested.

Entry from: IEECVET1.

Exit to:

- IEECVET4 if detect occurred in out-of-line display.
- IEECVETH/P/R/U (device dependent I/O modules) to process ENTER.
- IEECVFTA to process PFK line detect.
- IEECVET8 to process delete request.
- IEECVET9 to process verification of a delete operation.
- IEECVETD to issue error message.
- IEECVETE to issue error message.
- IEECVET1 to route ENTER for 3277 device.

IEECVETG — Open/Close Module (DIDOCS)

Operation: Performs open and close processing for cathode ray tube devices used as consoles.

Entry from:

- IEECVET1 if open pending.
- IEECVETK after it removes the console from roll mode.

Exit to:

- IEECLCTX if device was opened for console switch.
- IEECVET1 after open to perform other communications task operations.
- IEECVFTG to complete close of a console.
- IEECVETK to remove console from roll mode before close.

IEECVETH — 3066 Device I/O Module (DIDOCS)

Operation: Performs input/output operations to 3066 devices used as operator consoles.

Entry from: Any DIDOCS module that modifies the SIB.

Exit to:

- IEECVET1 after I/O has been initiated.
- IEECVFTG if status switch must take place.

IEECVETJ — Roll Mode Processor (DIDOCS)

Operation: Rolls messages off the console screen based on a timer interval and the number of lines to be rolled.

Entry from: IEECVET1.

Exit to:

- IEECVETK to display the number of messages waiting to be displayed.
- IEECVET2 to handle timer supervision if no roll occurred and to display messages if space exists in the message area.

IEECVETK — Timer Interpreter (DIDOCS)

Operation: Resolves the different time intervals at which roll is to occur for different CRT consoles.

Entry from:

- IEECVET1 if timer expires.
- IEECVETC to reset time interval after an asynchronous error.
- IEECVETG to remove a console from roll mode before closing the console.
- IEECVETA if roll mode or RTME (time interval) changes occur.
- IEECVFTR to set the timer.

Exit to:

- IEECVET2 to display message after roll.
- IEECVETE to display error message.
- IEECVETD to display warning message.
- IEECVETG to close the device.
- IEECVETH/P/R/U to write the screen image on the console screen.
- IEECVET1 if no roll is to occur.

IEECVETP — 2250 Device I/O Module - Part 1 (DIDOCS)

Operation: Performs I/O operations to 2250 devices used as operator consoles.

Entry from: Any DIDOCS module that modifies the SIB.

Exit to:

- IEECVET1 after I/O has been initiated.
- IEECVETF if light-pen or cursor detect occurs.
- IEECVFTG for a status switch.
- IEECVETQ to complete I/O operations.

IEECVETQ — 2250 Device I/O Module - Part 2 (DIDOCS)

Operation: Performs I/O operations to 2250 devices used as operator consoles.

Entry from: IEECVETP.

Exit to: IEECVET1 after I/O is initiated.

IEECVETR — 2260 Device I/O Module - Part 1 (DIDOCS)

Operation: Performs I/O operations to 2260 devices used as operator consoles.

Entry from: Any DIDOCS module that modifies the SIB.

Exit to:

- IEECVET1 after I/O is initiated
- IEECVFTR for status switch
- IEECVET4 if cancel occurs
- IEECVETF if cursor detect occurs
- IEECVFTR to complete I/O operations

IEECVETU — 3270 Device and Model 158 System Console I/O Module - Part 1 (DIDOCS)

Operation: Performs I/O operations to 3270 display devices used as operator consoles.

Entry from: Any DIDOCS module that modifies the SIB.

Exit to:

- IEECVET1 after I/O is initiated
- IEECVETF if light-pen or cursor detect occurs outside entry area
- IEECVFTG for a status switch
- IEECVETK for roll mode
- IEECVETV to complete I/O operations.

IEECVETV — 3270 Device I/O Module - Part 2 (DIDOCS)

Operation: Performs I/O operations to 3270 display devices used as operator consoles.

Entry from: IEECVETU.

Exit to: IEECVET1 after I/O has been initiated.

IEECVETW — 3284/3286 Console Device Support Processor

Operation: Performs the following functions for 3284/3286 printers used as a console:

- Opens the 3284/3286 as a console.
- Initiates a write operation to write messages to the console.
- After a write operation completes successfully, updates the console queue.
- When the 3284/3286 printer is to be removed from console status, closes the console.

Entry from: IEAVVCTR.

Exit to:

- IEAVSWCH if an I/O error occurs or the console cannot be opened.
- Issuer of SVC 72, otherwise.

IEECVET1 — DIDOCS Router Module

Operation: Passes control to DIDOCS modules on the basis of requested function.

Entry from: IEAVVCTR, IEECVETQ, IEECVETV, IEECVFTR, and IEECVFTB.

Exit to:

- IEECVETG for console open and close
- IEECVETC for asynchronous error processing
- IEECVET4 for command processing
- IEECVET2 for message output
- IEECVET7 for message deletion
- IEECVET9 for message deletion
- IEECVFTP for erasing a status display or stopping a dynamic display
- IEECVETJ for roll
- IEECVETK for timer processing
- IEECVFTM for out-of-line message output
- IEECVETF for light-pen or cursor detect
- IEECVETH/P/R/U (device-dependent I/O modules) to perform I/O
- IEECVFTA for PFK attention
- IEECVFTL for in-line message output
- IEECVFT1 for PFK redefinition
- IEECVFTT for display area blanking
- IEECVETD to issue error messages
- IEECVET6 for routed CONTROL E commands
- IEECVET8 for routed CONTROL E/CONTROL E,SEG commands
- IEECVETA for routed CONTROL S commands
- IEECVFTB for routed CONTROL D,PFK/CONTROL E,PFK/CONTROL N commands
- IEECVFTN for routed CONTROL D,F/CONTROL D,U/CONTROL D,H commands
- IEAVVCTR to return to the caller

IEECVET2 — In-line Message Output Module (DIDOCS)

Operation: Controls the output of in-line messages.

Entry from:

- IEECVFT2 to determine exit from message output modules after single-line message output.
- IEECVET1 to display a message.
- IEECVFTL to determine exit from message output modules after multiple-line message output.

- IEECVETJ to display a message after roll.
- IEECVETK to display a message after roll.
- IEECVET9 to display a message after automatic deletion occurs.

Exit to:

- IEECVETH/P/R/U (device-dependent I/O modules) to write messages to the screen.
- IEECVET1 if there were no messages to display.
- IEECVETD to issue a warning message.
- IEECVFT2 to move messages to the DCM.
- IEECVET7 to delete INTERVENTION REQUIRED messages.
- IEECVET9 for automatic deletion of messages.

IEECVET3 — Message Output Module (DIDOCS)

Operation: Issues STIMER for roll mode after roll has been initialized.

Entry from: IEECVET2 to issue STIMER for roll mode.

Exit to:

- IEECVET1 if no I/O is to be performed after STIMER.
- IEECVETD if 'unviewable message' warning line is requested.
- IEECVETH/P/R/U (device-dependent I/O modules) to write the screen image on the console screen.

IEECVET4 — Command Analyzer (DIDOCS)

Operation: Analyzes command input and routes the command to other DIDOCS modules or to the system for processing.

Entry from:

- IEECVET1 to process command input
- IEECVETF to process a cancel key or light-pen detect on a cancel function
- IEECVETR to perform the cancel function
- IEECVFTA to issue the command that PFK processing placed in the entry area
- IEECVETC to simulate pressing the CLEAR key

Exit to:

- IEECVETC to complete asynchronous error processing.
- IEECVET8 to initialize message deletion.
- IEECVET9 to process delete verification.
- IEECVET1 after command processing is complete.
- IEECVETH/P/R/U (device-dependent I/O modules) to rewrite the screen.
- IEECVETD for repeat last command processing.
- The appropriate CONTROL command processing modules (IEECVET6, IEECVET8, IEECVET9, IEECVETA, IEECVFTB, IEECVFTN, or IEECVFTP).

IEECVET6 — Message Deletion Module (DIDOCS)

Operation: Deletes messages requested by the K E,F or K E,nn(,nn) command.

Entry from: IEECVET1, IEECVET4.

Exit to:

- IEECVET9 to remove messages
- IEECVETD for message verification
- IEECVETE to issue error messages

IEECVET7 — Message Deletion Module (DIDOCS)

Operation: Deletes INTERVENTION REQUIRED messages and messages requested by DOM.

Entry from:

- IEECVET1
- IEECVET2
- IEECVFT2

Exit to:

- IEECVETH/P/R/U (device-dependent I/O modules) to write the screen
- IEECVET9 to attempt automatic deletion
- IEECVET1 on NOP entry

IEECVET8 — Message Deletion Module (DIDOCS)

Operation: Deletes messages requested by the K E,SEG command, a cursor detect, or a light-pen detect.

Entry from:

- IEECVETF to process a light-pen or cursor detect
- IEECVET4 to process the K E,SEG command
- IEECVET1 for routed K E/K E,SEG commands

Exit to:

- IEECVET9 to remove messages from the screen.
- IEECVETD to issue a verification message.
- IEECVETE to issue NO DELETABLE MESSAGE message.

IEECVET9 — Message Deletion Module (DIDOCS)

Operation: Processes automatic deletion when the screen is full, numbers messages in response to the K D,N command, and deletes messages in response to a K command.

Entry from:

- IEECVET4 to process the K D,N command.
- IEECVET6 to delete messages if no verification is necessary.
- IEECVET7 to perform automatic deletion.
- IEECVET2 to perform automatic deletion if screen is full.

Exit to:

- IEECVET2 after automatic deletion.
- IEECVETD to issue error messages.
- IEECVETH/P/R/U (device-dependent I/O modules) to rewrite the screen.

IEECVFTA — PFK-Entered Command Processor (DIDOCS)

Operation: Processes automatic command entry that results from a depressed PFK or a light-pen detect on the PFK line.

Entry from:

- IEECVET4 for depressed PFK
- IEECVETF for light-pen detect on the PFK line

Exit to:

- IEECVET4 to process an entered command (nonconversational mode).
- IEECVETH/P/R/U (device-dependent I/O modules) to write the command to the entry area of the screen (conversational mode).
- IEECVFTD to issue error messages.

IEECVFTB — PFK Definition Processor (K D,PFK/K E,PFK/K N,PFK) (DIDOCS)

Operation: Processes a PFK definition or redefinition in response to the K N,PFK command; displays or erases the PFK line in response to the K E,PFK or K D,PFK command.

Entry from: IEECVET1, IEECVET4.

Exit to:

- IEECVETH/P/R/U (device-dependent I/O modules) to write or erase the PFK line.
- IEECVFT1 when a PFK is defined.
- IEECVETD to write error messages.
- IEECVFTD to write error messages.

IEECVFTD — Message Module (DIDOCS)

Operation: Prepares for the writing of informational or error messages.

Entry from: IEECVFTA and IEECVFTB.

Exit to: IEECVETH/P/R/U (device-dependent I/O modules) to write messages to the screen.

IEECVFTG — DIDOCS Cleanup Module

Operation: Finishes functions that involve a system interface.

- If device is being closed, flags messages no longer needed.
- If device is changing status to message stream or full capability, flags out-of-line messages on the console queue.
- If device is changing status to status display, flags messages on console queue.
- If device is recovering from asynchronous error, flags out-of-line messages on the console queue.
- If this routine flags any out-of-line messages except for an asynchronous error, it stops a dynamic display, frees SACBs obtained with a GETMAIN, and resets the area definitions to their SYSGEN values.

Entry from:

- IEECVETG for open or close of a device
- IEECVETC for an asynchronous error
- IEECVETH/P/R/U (device-dependent I/O modules)
- IEECVFTR (device-dependent I/O module)

Exit to:

- IEECVET1 after close
- IEECVETC after all processing other than close

IEECVFTL — In-line Multiple-line Message Processor (DIDOCS)

Operation: Controls the output of in-line, multiple-line messages (MLWTO).

Entry from:

- IEECVET1
- IEECVFT2

Exit to:

- IEECVFT2 to process next WQE.
- IEECVET2 if roll mode is in effect and the screen is full.
- IEECVETH/P/R/U (device-dependent I/O modules) to write the screen.
- IEECVET1 if IEECVFTL encounters the end of the MLWTO chain but not the end of the message.
- IEECVETD to issue error messages.

IEECVFTM — Out-of-line Multiple-line Message Processor (DIDOCS)

Operation: Controls the output of out-of-line, multiple-line messages (MLWTO).

Entry from: IEECVET1.

Exit to:

- IEECVFTO for display lines that overlay messages.
- IEECVFTQ for display lines that do not overlay messages.
- IEECVET1 if no more out-of-line messages are on the queue.
- IEECVFTP for blanking of messages below a display.

IEECVFTN — Status Display Processor (DIDOCS)

Operation: Rewrites the display control line in response to a K D,F (framing), K D,H (holding), or K D,U (updating) command.

Entry from:

- IEECVET4 if console is a full-capability console.
- IEECVET1 if console is a status-display console.

Exit to: IEECVETH/P/R/U (device-dependent I/O modules) to write the screen.

IEECVFTO — Status Display Processor (DIDOCS)

Operation: Controls the output of a status display that overlays messages.

Entry from: IEECVFTM.

Exit to: IEECVETH/P/R/U (device-dependent I/O modules) to write three lines of the display area.

IEECVFTP — Status Display Processor (DIDOCS)

Operation: Controls the erasing of out-of-line displays.

Entry from: IEECVET1, IEECVET4.

Exit to:

- IEECVETH/P/R/U (device-dependent I/O modules) to write the console screen.
- IEECVFTT to blank the screen below the erased display.
- IEECVET1 for a NOP entry.

IEECVFTQ — Status Display Processor (DIDOCS)

Operation: Controls the output of a status display to an out-of-line area that does not overlay messages.

Entry from: IEECVFTM.

Exit to:

- IEECVETH/P/R/U (device-dependent I/O modules) to write display lines to the screen.
- IEECVFTM if no I/O is done.

IEECVFTR — 2260 I/O Module - Part 2 (DIDOCS)

Operation: Performs I/O operations to 2260 devices used as operator consoles.

Entry from: IEECVETR.

Exit to:

IEECVFTG for cleanup.
IEECVETK for setting the timer.
IEECVETR for additional I/O processing.
IEECVET1 if a status switch is not required.

IEECVFTT — Status Display Processor (DIDOCS)

Operation: Overlays with blanks any in-line message that is below an out-of-line display or between two out-of-line displays.

Entry from:

- IEECVET1
- IEECVFTM

Exit to: IEECVETH/P/R/U (device-dependent I/O modules) to write the screen.

IEECVFT1 — PFK Definition Processor (DIDOCS)

Operation: Updates PFK definitions in SYS1.DCMLIB.

Entry from: IEECVET1.

Exit to:

- IEECVET1 after I/O to SYS1.DCMLIB has been initiated.
- IEECVETH/P/R/U (device-dependent I/O modules) to blank the entry area.

IEECVFT2 — Single-line Message Processor (DIDOCS)

Operation: Displays and marks both action and deletable messages; locates and routes nondisplayed multiple-line messages (MLWTOs).

Entry from:

- IEECVET2 to move messages to the DCM.
- IEECVFTL to process the next WQE.

Exit to:

- IEECVET2 to continue output queue scan.
- IEECVFTL to process in-line, multiple-line messages.

IEEC2740 — 2740 Console Device Support Processor

Operation: Performs the following functions for a 2740 communications terminal used as a console.

- Opens the 2740 communications terminal as a console.
- Uses BTAM to read a command from the console.

- After a read operation completes successfully, passes the command to command processing (SVC 34).
- Uses BTAM to write messages to the console.
- When the 2740 communications terminal is to be removed from console status, closes the console.

Entry from: IEAVVCTR.

Exit to:

- IEAVSWCH if an I/O error occurs or the console cannot be opened.
- Otherwise IEAVVCTR.

IEEDISPD — Display Domain Processor

Operation: Writes a console display of entries in the Domain Descriptor Table.

Entry from: IEEVWAIT.

Exit to: End of Task.

IEEJB840 — See IGC0203E

IEEMB803 — System Log Initialization/Writer Module

Operation: This module initializes the system log, opens and closes the log data set, and writes the log buffers to the log data set.

Entry from: IEEVWAIT.

Exit to: None - always available through waiting on ECB.

IEEMB804 — Write-To-Log (WTL) Processor — SVC 36

Operation: This module processes all valid WTL requests.

Entry from: IEE1603D.

Exit to: IEE1603D.

IEEMB805 — System Log Resource Initialization

Operation: This module acquires and initializes log resources used through log activation.

Entry from: IEEMB803.

Exit to: IEEMB803.

IEEMB806 — System Log ESTAE Processor

Operation: This module sets up two ESTAE environments for the system log, and it processes system log task ABEND situations. (The second ESTAE handles abnormal terminations of the first ESTAE).

Entry from: IEEMB803 (ESTAE/ABEND interface).

Exit to: ESTAE/ABEND interface.

IEEMB807 — System Log Message Module

Operation: This module issues messages for the system log task.

Entry from: IEEMB803, IEEMB806.

Exit to: Caller.

IEEMB808 — Master Trace Data Entry Routine

Operation: This module adds entries to the master trace table.

Entry from: IEETRACE macro processing.

Exit to: Caller.

IEEMB809 — Master Trace Create/Deactivate Routine

Operation: This module initializes a new master trace table or deactivates an existing master trace table.

Entry from: IEETRACE macro processing.

Exit to: Caller.

IEEMB810 — Performance Group Reset Module

Operation: This module passes performance group reset parameters to the system resources manager (SRM).

Entry from: IEEVWAIT.

Exit to: End of task.

IEEMB811 — SET Keyword Scanner

Operation: This module scans for various SET keywords and passes control to the appropriate keyword processor.

Entry from: IEEVWAIT.

Exit to: End of task.

IEEMB812 — SET IPS, Installation Control Specification, OPT Processor

Operation: This module invokes a list processor to process the data in the appropriate parmlib member (IEAIPSxx, IEAICSxx, or IEAOPTxx). It also notifies the system resources manager of the changes.

Entry from: IEEMB811.

Exit to: IEEMB811.

IEEMB813 — UNLOAD Command Syntax Scanner

Operation: This module scans the information in the UNLOAD command and validates the syntax.

Entry from: IEEVWAIT.

Exit to: End of task.

IEEMB814 — Message Module for Parmlib Process

Operation: This module issues messages for IEEMB811 and IEEMB878.

Entry from: IEEMB811 and IEEMB878.

Exit to: Caller.

IEEMB815 — CHNGDUMP Command Processor

Operation: This module implements the CHNGDUMP operator command. IEEMB815 scans the operand portion of the command and updates the RTCT dump options lists. It fixes pages needed for the SUMDUMP option of SVC dump. If the NOSUMDUMP option is specified on the CHNGDUMP command, it frees pages.

Entry from: IEE0403D via BALR.

Exit to: IEE0003D.

IEEMB816 — Master Trace Functional Recovery Routine

Operation: This module performs error recovery for master trace routines IEEMB808 and IEEMB809.

Entry from: RTM via FRR stack.

Exit to: Caller of master trace.

IEEMB820 — SMF Initialization Router Module

Operation: This module performs the following functions:

- Initializes SMCA/STAE/ACT
- Invokes SMF parameter processing
- Initializes SMF recording

Entry from: IEEMB860.

Exit to: Caller.

Called routines: IEEMB821, IEEMB822, IEEMB823, IEEMB824 via BALR.

IEEMB821 — SMF Input Merge Module

Operation: This module processes the SYS1.PARMLIB member containing SMF options. It parses the values, prompts the operator for changes, and sets the resulting values in the SMCA.

Entry from: IEECB913, IEEMB820, IEEMB835.

Exit to: Caller.

Called routines: IEEMB832, IEEMB833, IEEMB824, IEEMB839 (ELEMENQ routine), IEEMB822 via BALR.

IEEMB822 — SMF Data Set Initialization

Operation: This module prepares SMF data sets for recording by:

- Dynamically allocating the data sets specified in the data set parameter
- Pre-formatting the data sets, if necessary
- Creating SMF buffers

Entry from: IEEMB820 at IPL time; IEEMB835 at SET SMF time.

Exit to: Caller.

Called routines: IEEMB824 via BALR.

IEEMB823 — SMF Initialization Record Processor

Operation: At IPL time, this module issues the four initialization record types (0, 8, 22, 90). It issues SVC 78 to produce a fifth record (type 19). It completes SMF initialization for the master address space. Additionally, it prompts the operator for the IPL reason.

At SET SMF and SETSMF time, this module produces a type 90 record containing the new SMF options. At SET SMF time, it also reinitializes SMF for the master address space.

Entry from: IEEMB820 at IPL time; IEEMB835 at SET SMF time to build record type 90; IEECB913 at SET SMF time to build record type 90.

Exit to: Caller.

Called routines: IEEMB824 via BALR; IEFSMFIE via LINK.

IEEMB824 — SMF Message Processor Module

Operation: This module builds and issues messages according to the parameters passed it by the caller. Messages issued are WTOs, WTORs, multi-line WTOs, and PUTs to a message data set.

Entry from: IEECB913, IEECB916, IEEMB820, IEEMB821, IEEMB822, IEEMB823, IEEMB825, IEEMB827, IEEMB829, IEEMB831, IEEMB832, IEEMB833, IEEMB835, and IFASMFDP.

Exit to: Caller.

Called routines: References IEEMB826.

IEEMB825 — SMF ESTAE Exit Routine

Operation: If the SMF writer abends, this module performs recovery. Its functions include dump processing, message processing, and retry or SMF recording cleanup.

Entry from: RTM.

Exit to: Caller.

Called routines: IEEMB824, IEEMB827 (global cleanup subroutine).

IEEMB826 — SMF Message Processor Language Parts Table

Operation: This module contains translatable language parts which are used by IEEMB824 to build SMF messages.

Entry from: Not applicable (non-executable code).

Exit to: Not applicable (non-executable code).

IEEMB827 — SMF Initialization ESTAE

Operation: This module manages recovery from failures that occur during SMF initialization. Specifically, it sends messages to the operator, creates an SDUMP, and cleans up resources.

Entry from: RTM.

Exit to: Caller.

Called routines: IEEMB824.

IEEMB828 — SMF Message Processor Table

Operation: This dummy CSECT is part of the SMF message processor, IEEMB824. It contains a series of indexes used by IEEMB824 to locate messages.

Entry from: Not applicable (non-executable code).

Exit to: Not applicable (non-executable code).

IEEMB829 — SMF Writer Task

Operation: This module completes SMF initialization and then handles those parts of SMF recording that require TCB mode:

- Completing the data set switch
- Ensuring that SMF data sets were dumped by the operator as requested
- Performing part of the processing for the SET SMF command
- Handling I/O errors

Entry from: IEECB913 via BALR, IEEMB860, via ATTACH.

Exit to: Caller.

Called routines: IEEMB824, IEEMB839 via BALR.

IEEMB830 — SMF Writer SVC

Operation: This module places SMF records into the SMF buffers (VSAM control intervals).

Entry from:

For entry point IEEMB830: Issuer of a SMFEWTM macro.

For entry point IGC0008C: SVC FLIH.

Exit to: Caller.

Called routines: IEASMFSP via BALR.

IEEMB831 — SMF Syntax Analyzer Module

Operation: This module is used by IEEMB832 to analyze input text and return a number corresponding to the data description. It also processes syntax and invalid option errors.

Entry from: IEEMB832.

Exit to: Caller.

Called routines: IEEMB824 via BALR.

IEEMB832 — SMF Queue Options Module

Operation: This module parses the input text. If no errors are found, it adds the option to the SMF option chain.

Entry from: IEEMB821 or IFASMFDP.

Exit to: Caller.

Called routines: IEEMB824, IEEMB831 via BALR.

IEEMB833 — SMF List Options Module

Operation: This module lists the SMF options currently in effect. The list is directed to the console in the case of IPL or SET; to SYSPRINT in the case of the dump program (IFASMFDP).

Entry from: IEEMB821, IFASMFDP, or IEECB916.

Exit to: Caller.

Called routines: IEEMB824 via BALR.

IEEMB834 — SMF Writer SRB

Operation: This module takes the VSAM control intervals from the buffers and writes them out to the SMF data set.

Entry from: Dispatcher (IEAVEDS0).

Exit to: Caller.

Called routines: IEASMFSP via BALR.

IEEMB835 — SMF SET Command Processing

Operation: This module processes the SYS1.PARMLIB member specified by a SET command. It parses the new values and resets the SMCA values. If data sets are changed, it opens and closes the appropriate data sets.

Entry from: IEEMB811.

Exit to: IEEMB811.

Called routines: IEEMB821, IEEMB822, IEEMB823, IEEMB824, IEFJSREQ.

IEEMB836 — Interval Data Collection Module

Operation: This module is scheduled when the SMF interval timer for a job expires. Running in SRB mode, it builds and writes interval records (types 30 and 32).

Entry from: Dispatcher (IEAVEDS0).

Exit to: Caller.

Called routines: IEFTB727, IEFTB728.

IEEMB837 — SMF Option Tables

Operation: This module consists of the following CSECTS:

- SMFOPTAB - A table of valid keywords
- SMFDEFLT - A table of SMF writer default options
- DMPDEFLT - A table of default options for the SMF dump program (IFASMFDP)

Entry from: Loaded by IEEMB820; referenced in IEEMB821 and IFASMFDP.

Exit to: Not applicable (non-executable code).

IEEMB838 — SMF Macro Module

Operation: This module performs the processing invoked by the SMF macros.

Entry from: SMF macros: SMFEWTM, SMFRTEST, SMFDETAL, SMFINTVL, and SMFEXIT.

Exit to: Caller.

IEEMB839 — SMF Timer Program

Operation: This module is branch entered by the caller to enqueue or dequeue timer elements. The module handles four types of processing: MAXDORM, STATUS, INTERVAL for each ASID, and dump detector.

Entry from: IEEMB821, IEEMB829, IEEMB836, IEEMB842.

Exit to: Caller.

IEEMB842 — SMF MAXDORM and STATUS Processor

Operation: This module performs STATUS and MAXDORM processing. For STATUS, it builds and writes the type 23 statistics record. For MAXDORM, if the current record has been in the SMF buffer longer than the interval specified by the MAXDORM keyword in the SMFPRMxx parmlib member, IEEMB842 schedules the SMF writer SRB to force the data to be written to the SMF data set.

Entry from: Scheduled by IEEMB839 at the expiration of the MAXDORM and STATUS time intervals.

Exit to: Caller.

IEEMB846 — TSO Command Table for Accounting

Operation: This module includes all commands counted in the SMF type 32 record.

Entry from: Not applicable (non-executable code).

Exit to: Not applicable (non-executable code).

IEEMB847 — SMF Message Language Part Table

Operation: This dummy CSECT is contained in the SMF message processor language parts table, IEEMB826.

Entry from: Not applicable (non-executable code).

Exit to: Not applicable (non-executable code).

IEEMB848 — SMF Dynamic DD Data Collector Module

Operation: This module saves EXCP counts (1) for data sets dynamically unallocated and (2) for all data sets at dynamic deconcatenation/ concatenation. The counts are then available for the SMF type 30 record.

Entry from: IEFDB4F9.

Exit to: Caller.

IEEMB860 — Master Region Initialization Routine

See *OS/VS2 System Initialization Logic* for a module description of IEEMB860.

IEEMB876 — SET Command Keyword Table

Operation: This module contains the SET command keywords processed by IEEMB811.

Entry from: IEEMB811.

Exit to: IEEMB811.

IEEMB877 — Parmlib Variable Messages

Operation: This module contains the messages for parmlib processing.

Entry from: IEEMB814.

Exit to: IEEMB814.

IEEMB878 — SYS1.PARMLIB Read Routine

Operation: This module reads records from a specified member of SYS1.PARMLIB.

Entry from: IEECB805, IEEMPDEV, IEFJSIN2.

Exit to: Caller.

IEEMB879 — Master Trace SVC Dump Exit

Operation: This module is invoked by SVC dump and uses SVC dump services to dump the master trace table and its related control blocks. It schedules an SRB (IEEMB880) to run in ASID 1 and moves the master trace data to a common area buffer. It invokes the SVC dump exit output service (IEAVTSEO entry in IEAVTSDO) to write the master trace data on the dump data set.

Entry from: IEAVTSDU via BALR.

Exit to: Caller.

IEEMB880 — Master Trace Dump Exit Data Move Routine

Operation: This module runs as an SRB in the address space containing the master trace table data. It is scheduled by IEEMB879 for every page of master trace data to be dumped. It moves one page (4K) of master trace data to a common buffer and then posts the ECB (via IEAVSY50) that IEEMB879 is waiting on.

Entry from: Dispatcher (IEAVEDS0) as an SRB scheduled by IEEMB879.

Exit to: IEAVEDS0.

IEEMB881 — System Address Space Create Routine

Operation: This module creates a system address space either before or after master scheduler initialization is complete. The address space can be created with a limited-function start or a full-function start. If it is created with a full-function start, IEEMB881 passes the START parameters (input to IEEMB881) to the START command syntax check routine (IEEVSTAR).

Entry from: IEEVIPL, a resource initialization module (RIM), or any other module in supervisor mode.

Exit to: Caller.

Called routines: IEAVEMCR, IEAVEMRQ, IEEVWAIT.

Entry point EAESTAE: This routine receives control if an ABEND occurs during system address space create processing. It fills in the SDWA (if one exists), takes a dump, and passes control to RTM, which attempts a retry at entry point EARETRY.

Entry from: RTM.

Exit to: RTM.

Entry point EARETRY: This entry point sets the return and reason codes, releases storage for the CSCB, terminates the address space (if one was created), and terminates the ESTAE.

Entry from: RTM.

Exit to: Caller of IEEMB881.

IEEMB882 — Positional Parser Routine

Operation: This module syntax checks a record using a rules list associated with the record. The rules list specifies the number of parameters in the record along with the minimum and maximum length of each parameter. It also indicates if the parameter is optional and if it can be enclosed in apostrophes.

Entry from: IEECB805, IEEJSIN2.

Exit to: Caller.

IEEMB883 — System Address Space Initialization WAIT/POST Routine

Operation: This module waits for an event (indicated by the event code passed as input) to complete. If event code 1 is specified, IEEMB883 waits for master scheduler initialization to complete.

Entry from: IATINXM, IEAVTSAI, IEAVXMAS, IEEPRWI2, IEFAB4I1, ISGNASIM.

Exit to: Caller.

Entry point WPESTAE: This entry point receives control if an abend occurs during IEEMB883. It fills in the SDWA (if one exists) and takes a dump.

Entry from: RTM.

Exit to: RTM.

Entry point WPRETRY: This entry point receives control from RTM following the execution of WPESTAE. It sets return and reason codes.

Entry from: RTM.

Exit to: Caller of IEEMB883.

IEEMB884 — Scheduled Commands Table

Operation: This module contains a table consisting of the verb code, module name of the command processor, and the target execution address space for each scheduled command.

Entry from: N/A.

Exit to: N/A.

IEEMPDEV — Display Deviation Processor

Operation: This module compares the configuration described in the CONFIXxx member of SYS1.PARMLIB with the actual system configuration and displays any differences to the operator.

Entry form: IEEMPDM.

Exit to: Caller.

IEEMPDM — DISPLAY Matrix Command Processor

Operation: This module displays, to the operator, the following information: status of processor, channel, and device paths; high storage address; storage status (on/off line).

Entry from: IEEVWAIT.

Exit to: Caller.

IEEMPS03 — QUIESCE Command Processor

Operation: This module suspends system activity by calling the load-wait-state processor (IEEVLDWT).

Entry from: IEEVWAIT.

Exit to: Caller.

IEEMPVST — VARY Storage Processor

Operation: This routine either makes a contiguous range of 4K blocks of storage available online or takes a range of blocks offline.

Entry from: IEEVWAIT via ATTACH, IEEVSTOR.

Exit to: Caller.

IEEPALTR — SVC 110 Interface

Operation: This routine issues SVC 110 for DISPLAY U, DISPLAY C,K, or DISPLAY PKF commands.

Entry from: IEEVWAIT via ATTACH.

Exit to: Caller.

IEEPRTN2 — STC Free Region Routine

Operation: This module determines if there is a CSCB in existence at the end of started task control processing. If there is, IEEPRTN2 frees the CSCB.

Entry from: IEESB605, IEEPRWI2, IEEVMNT1, IEEVSTAR, or IKJEFLA via XCTL.

Exit to: Region control task via SVC 3.

IEEPRWI2 — STC Get Region Routine

Operation: This module gets a new region for started task control processing. If a system component address space is being created, this routine invokes the address space initialization routine. It also determines whether a START, MOUNT, or LOGON command is being processed, and based on that decision invokes the appropriate command processor.

Entry from: IEAVAR00 by means of ATTACH.

Exit to: IEEVSTAR, IEEVMNT1, IKJEFLA, or IEEPRTN2.

Called routines: IEEMB883 via BALR, a system address space initialization routine via LINK, IFAESIL via LINK.

IEESB601 — SWA and TIOT Initialization for Private Catalog Processing

Operation: This module creates an SWA control block structure for use by the dynamic allocation routines. These routines are invoked to process private catalogs for START, MOUNT, and LOGON commands.

Entry from: IEESB601 via BALR.

Exit to: IEESB601 via branch.

IEESB605 — Job Scheduling Subroutine

Operation: This module sets up the environment necessary to initiate a START, MOUNT, or LOGON command task. It builds the SSOB, SSIB, local job

scheduling parameter lists, and an SWA structure for the task; then it invokes the initiator. When initiator processing is done, IEESB605 deletes the SWA structure it created and frees the CSCB for the task.

Entry from: IEEVJCL via XCTL for a START or MOUNT command, or IKJEFLB for LOGON commands.

Exit to: IEFSD060, an initiator routine via LINK, to initiate the command task; IEEPRTN2 via XCTL, for end processing of a START or MOUNT command; or IEEPRTN2 via XCTL, for end processing of LOGON commands.

IEESB606 — STC Get JSCB Routine

Operation: This routine builds a JSCB during storage initialization for a START, MOUNT, or LOGON command.

Entry from: IEAVEMIN via CALL.

Exit to: Caller.

IEESB665 — STC Recovery Exit Routine

Operation: This module schedules a retry of STC when one of the following conditions has occurred:

- A program check
- A machine check
- An ABEND
- Depression of the RESTART key on a console

If percolation from a lower level recovery routine has occurred, IEESB665 continues percolation.

Entry from: Recovery termination management via LINK.

Exit to: Caller via branch.

IEESB670 — Job Scheduling Subroutine Recovery Exit Routine

Operation: This module schedules a retry of IEESB605 when one of the following conditions has occurred:

- A program check
- A machine check
- An ABEND
- Depression of the RESTART key on a console

If percolation from a lower level recovery routine has occurred, IEESB665 continues percolation.

Entry from: Recovery termination management via LINK.

Exit to: Caller via branch.

IEESTPRS — Stop/Restart Subroutine

Operation: This module stops the operating system in a manner that allows a restart interruption to bring the system back into operation. This module runs on one processor and stops all others.

Entry from: IEEVLDWT and any routine in supervisor state that wants to put the system into a restartable state.

Exit to: Caller.

IEEVALST — Validate Page Routine

Operation: This module sets to zero the storage and storage key of a 4K section of real storage.

Entry from: IEEMPVST, IEEVSTGL.

Exit to: Caller.

IEEVCPR — Processor/Channel Reconfiguration Processor

Operation: IEEVCPR physically and logically adds or removes a processor and/or channel(s) to or from the system.

Called by: IEEVCPU.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEEVCPRL, IEECLEAN, and IEEVMESS.

IEEVCPRL — Processor/Channel Logical Reconfiguration Module

Operation: IEEVCPRL performs the functions required to make a processor and/or channel(s) logically available or unavailable to the system.

Called by: IEEVCPR.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: AHLVCON, AHLVCOFF, IEAVRTOD, IEAVRSSC, IEAVSPSA, IEAVTSIN, IECVCINT, IECVCRHA, IECVCRHD, IECVCRHV, IECVIOPM, IEEVDEV, IEEVMESS, IEEVWKUP, IEFAB49C, IEFENFFX, IGC0009A, IGC0407B, or IGFPBUCR.

IEEVCPU — VARY CPU/CH Command Processor

Operation: IEEVCPU controls VARY CPU/CH command processing. It parses the command, calls IEEVCPR to perform the requested reconfiguration, and writes the reconfiguration messages after IEEVCPR returns control.

Called by: The master scheduler (IEEVWAIT) via an ATTACH macro after the operator issues a VARY CPU/CH command.

Exit to: Caller.

Error exit: RTM.

Called routines: IEEVMESS and IEEVCPR.

IEEVDCCR — Disabled Console Communication Routine

Operation: This module enables disabled recovery routines to communicate with the system operator through the master console (or its alternate) during system recovery processing.

Entry from: IEESTPRS, IEEVEXSN, IGFPXMFA.

Exit to: Caller.

IEEVDEV — Device Subroutine

Operation: This routine obtains information regarding the condition of available paths to a device and provides that information to the caller.

Entry from: Allocation, DDR, VAP, VARY CPU, VARY PATH, IEEVCPRL.

Exit to: Caller.

IEEVEXSN — Excessive Spin Notification Routine

Operation: This routine is called by various system spin routines to communicate with the operator when an excessively long spin is detected. The operator is asked if the spin should continue or if an ACR condition for the other processor should be simulated.

Entry from: IEAVELK, IEAVERI, IEEVLDWT, IEAVTMTC, and IEAVINV.

Exit to: Caller.

IEEVIPL — Master Scheduler Base Initialization

See *OS/VS2 System Initialization Logic* for a module description of IEEVIPL.

IEEVJCL — Job Control Language Build Routine

Operation: This module builds internal JCL text for a command task that represents a START, MOUNT, or LOGON command.

Entry from: IEEVSTAR via branch for a START command; IEEVMNT1 via branch for a MOUNT command; IKJEFLA via branch for a LOGON command.

Exit to: IEESB605 via XCTL.

IEEVLDWT — Load Wait State Processor

Operation: This module serves as the interface for loading a disabled wait state or restartable wait state. For restartable wait states, it spins for the restart resource and interfaces with the stop/restart subroutine.

Entry from: IEEMPS03, IECVIRST, and IECVRSTI.

Exit to: Caller.

IEEVMESS — Reconfiguration Message Module

Operation: IEEVMESS performs message processing for all the reconfiguration modules. It is called to:

- Obtain storage for and initialize a message buffer
- Place a specified message in a given message buffer
- Write out the messages in a specified message buffer

- Issue a specified message as a WTO or WTOR
- Free message buffer storage

Called by: Reconfiguration modules.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEEVTEXT.

IEEVMNT1 — MOUNT Command Syntax Check Routine

Operation: This module checks a MOUNT command and its parameters for correct syntax. It also uses this input to build a START descriptor table (SDT) that contains internal JCL.

Entry from: IEEPRWI2 via XCTL.

Exit to: IEEVJCL via branch for normal processing; IEEPRTN2 via XCTL for error processing.

IEEVMNT2 — MOUNT Command Processor

Operation: This module checks the unit control block (UCB) associated with the device to be mounted. If the device is a tape drive marked reserved, IEEVMNT2 issues an error message. Otherwise, it also checks the MOUNT command input buffer (CIB) for the use attribute specified for the device.

Entry from: IEFSD263, an initiator routine, via ATTACH.

Exit to: IEFSD263 via branch.

IEEVMSG — STC Message Module

Operation: This module contains the text of messages issued by the started task control routines.

Entry from: IEEVMNT2 or IEESB605 via CALL.

Exit to: Caller via branch.

IEEVPTH — VARY Path Command Processor

Operation: This routine makes individual paths to a device either available or unavailable for I/O processing.

Entry from: IEEVWAIT.

Exit to: Caller.

IEEVSEND — Operator SEND Command Main Control

Operation: This module provides a console or TSO-terminal operator with a means of communicating with other TSO and/or console operators.

Entry from: IEEVWAIT.

Exit to: End of task.

IEEVSND2 — SEND Command Mail Handler

Operation: This module saves the mail for specified users in the mail section of the SYS1.BRODCAST data set.

Entry from: IEEVSND9.

Exit to: IEEVSND9.

IEEVSND3 — SEND Command List Handler

Operation: This module lists all notices from or recovers a specific notice from, the SYS1.BRODCAST data set for the sending operator.

Entry from: IEEVSND9.

Exit to: IEEVSND9.

IEEVSND4 — SEND Command Cleanup Routine

Operation: This module issues informational, warning, and error messages to the operator.

Entry from: IEEVSEND.

Exit to: IEEVSEND.

IEEVSND5 — Operator SEND Command I/O Routine

Operation: This module performs I/O operations for the SEND command.

Entry from: IEEVSND2, IEEVSND3, IEEVSND8, IEEVSND9.

Exit to: Caller.

IEEVSND6 — SEND Command Now Processor

Operation: This module issues the SEND command text to the specified recipients.

Entry from: IEEVSEND.

Exit to: IEEVSEND.

IEEVSND8 — SEND Command Notice Handler

Operation: This module adds notices to or deletes notices from the SYS1.BRODCAST data set.

Entry from: IEEVSND9.

Exit to: IEEVSND9.

IEEVSND9 — SYS1.BRODCAST Data Set Access Controller

Operation: This routine coordinates the operator usage of the SYS1.BRODCAST data set.

Entry from: IEEVSEND.

Exit to: IEEVSEND.

IEEVSTAR — START Command Syntax Check Routine

Operation: This module checks a START command and its parameters for correct syntax. It uses this input to build a START descriptor table (SDT) that contains the internal JCL for the task to be started.

Entry from: IEEPRWI2 via XCTL.

Exit to: IEEVJCL via branch, for normal processing; IEEPRTN2 via XCTL, for error processing.

IEEVSTEE — ESTAE Routine for IEEVSTEL

Operation: This module performs recovery processing for IEEVSTEL. Unless a previous recovery routine has already produced diagnostic information, IEEVSTEE writes a record in SYS1.LOGREC and issues an SDUMP.

Called by: RTM.

Exit to: Caller.

IEEVSTEL — Vary Storage Element Reconfiguration Processor

Operation: IEEVSTEL varies a storage element on or offline in response to a VARY STOR command.

Called by: IEEVSTOR.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEEVSTGL, IEEVSTFA, and IEEVMESS.

IEEVSTFA — Find Alternates for Vary Storage Element Offline

Operation: When varying a storage element offline, this module determines which, if any, R-S pairs in the storage element contain MVS preferred data or hardware system area. For each R-S pair that does, IEEVSTFA selects from another storage element a suitable R-S pair into which the data can later be swapped.

Called by: IEEVSTEL.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEAVRCF3 and IEEVSTGL.

IEEVSTGL — Vary Storage Logical Routine

Operation: IEEVSTGL is called during VARY STOR command processing to logically vary a specified number of frames on or offline.

Called by: IEEVSTGP, IEEVSTEL, and IEEVSTFA.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEAVRCF3, IEEVMESS, and IEEVALST.

IEEVSTGP — Vary Storage Physical Routine

Operation: IEEVSTGP varies a range or an amount of storage on or offline in response to a VARY STOR command.

Called by: IEEVSTOR.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEAVRCF3, IEEVMESS, and IEEVSTGL.

IEEVSTOP — Vary CPU Stop Routine

Operation: IEEVSTOP stops the processor on which it is running. In order to stop the processor, IEEVSTOP ensures that the CSD indicates that the processor is offline, resets the processor's prefix register to zero, and issues a SIGP STOP instruction to the processor.

Entry from: Machine check handler, timer, vary CPU offline.

Exit to: None.

IEEVSTOR — VARY STOR Command Processor

Operation: IEEVSTOR controls VARY STOR command processing. It:

- Determines if the command is valid and has been issued from an authorized source
- Routes control to the proper routine to perform the reconfiguration
- Issues the appropriate reconfiguration message(s) after the storage has been varied

Called by: The master scheduler (IEEVWAIT) via an ATTACH macro after the operator issues a VARY STOR command.

Exit to: Caller.

Error exit: RTM via an ABEND macro.

Called routines: IEEMPVST, IEEVMESS, IEEVSTEL, and IEEVSTGP.

IEEVSTPE — ESTAE Routine for IEEVSTGP

Operation: IEEVSTPE performs recovery processing for IEEVSTGP. Unless diagnostic information has already been produced, IEEVSTPE writes a record in SYS1.LOGREC and issues an SDUMP. It also issues a message telling the operator what happened before the error occurred. In some cases, it backs out of processing done prior to the error.

Called by: RTM.

Exit to: Caller.

Called routines: IEAVRCF3, IEEVMESS, and IEEVSTGL.

IEEVSTRE — ESTAE Routine for IEEVSTOR

Operation: IEEVSTRE performs recovery processing for IEEVSTOR. It records the diagnostic information in SYS1.LOGREC, issues an SVC dump, and issues message IEE713I.

Called by: RTM.

Exit to: Caller.

Called routines: IEEVMESS.

IEEVWAIT — Master Scheduler Wait

In the Master Scheduler Address Space

Operation: At system initialization time, IEEVWAIT terminates system trace (if necessary), and attaches the system log task. After system initialization, IEEVWAIT's main function is to scan the CSCB chain and process the pending commands that execute in the master scheduler address space. If invoked for a START command, IEEVWAIT posts ECB CHASWT to notify the caller when master scheduler initialization is complete.

Entry from: IEEMB860 at system initialization time; IEE0803D via POST.

Exit to: No normal exit. This is a permanent task.

Error entry: At entry point STAE0000 from failure or from ABEND.

Error exit: To a wait state if restart fails.

In the Communications Task Address Space

Operation: During system initialization, IEEVWAIT performs no major function in the communications task address space. After system initialization, IEEVWAIT's only function in the communications task address space is to scan the CSCB chain and process

the pending commands that execute in this address space.

Entry from: IEAVN701 during system initialization, IEE0803D via POST.

Exit to: No normal exit. This is a permanent task.

Error entry: At entry point STAE0000 from failure or from ABEND.

Error exit: To the dispatcher.

IEEVWKUP — Vary CPU Wakeup and Quiet Routine(s)

Operation: This routine performs one of three functions:

- It completes the preliminary initialization for the processor that is being brought online.
- It performs a quiesce function for the vary offline process. This includes resetting control registers to their initial values.
- It initializes the channel availability table in the physical configuration communication area of the executing processor.

Entry from: IEEVCPRL.

Exit to: Dispatcher (when varying a processor online), IEEVSTOP (when varying a processor offline).

IEEXEDNA — Display Consoles Command Processor

Operation: This module writes a console status display to the console that issued the command.

For input stream commands, the display goes to the master console. If L=cca is specified, output goes to the routed console, which may be different from the one issuing the command.

Entry from: IEEVWAIT.

Exit to: Dispatcher or end of task.

IEE0003D — ESTAE Environment Creation Routine

Operation: This routine creates an ESTAE environment routine to protect the command scheduler from ABEND situations.

Entry from: Issuer of SVC 34.

Exit to: IEE0303D.

IEE00110 — Master Scheduler SVC 110 Router

Operation: This module routes control for the processing of the commands D C,K; D U; and D PFK.

Entry from: IEEPALTR.

Exit to: IEE10110, IEE20110, IEE40110.

IEE0303D — Control Block Chain Manipulator; QEDIT and MGCR Macro Function Processor

Operation: This module manipulates chains. It also communicates ABTERM requests to the ABTERM routines, and it is the QEDIT processor.

Entry from: IEE0003D.

Exit to: IEE5403D.

IEE0403D — Command Scheduler Router

Operation: This module scans, identifies, and verifies commands. It gives control to the proper command processor.

Entry from: IEE5403D.

Exit to: IEE0503D, command processors.

IEE0503D — Command Processing Message Assembly

Operation: This module assembles and edits messages for command processors. It issues WTO or TPUT macros to write the messages.

Entry from: Command processors.

Exit to: The address passed by the caller.

IEE0603D — Immediate Routine for SET Command

Operation: This module inspects the operands of the SET command and routes control to the appropriate processing module.

Entry from: IEE0403D.

Exit to: IEE6503D, IEE6603D, IEE0803D, IEE0503D.

IEE0703D — MODIFY and STOP Command Processor

Operation: This module performs processing for the STOP and MODIFY commands.

Entry from: IEE0403D.

Exit to: IEE0003D.

IEE0803D — CSCB and ASCB Creation

Operation: For task-creating commands, this module builds a CSCB. For address space-creating commands, this module interfaces with address space-requesting routines.

Entry from: Variable.

Exit to: IEE0003D, to post IEEVWAIT.

IEE10110 — Display C, K Processor

Operation: This module writes a status display showing the CONTROL (K) command operands using a multiple-line WTO macro.

Entry from: IEE00110 via CALL.

Exit to: IEE00110.

IEE1403D — HALT, SWITCH, and TRACE Command Syntax Checker

Operation: This module preprocesses the HALT, SWITCH, and TRACE commands by checking the command syntax.

Entry from: IEE0403D.

Exit to: IEE0803D, IED1303D, ISTCFF3D.

IEE1603D — LOG and WRITELOG Command Processor

Operation: This module processes the commands LOG and WRITELOG (with the options CLOSE, START, or CLASS).

Entry from: IEE0403D.

Exit to: IEE0003D.

IEE20110 — Display Unit Processor — Unit Status, One

Operation: This module analyzes the operands of a D U command, gets a work area, and finds the first UCB for the first unit to be displayed.

Entry from: IEE00110.

Exit to: IEE23110, IEE22110.

IEE21110 — Display Unit Processor — Unit Status, Two

Operation: This module loads the device name table the first time it is needed and creates the text of each line to be displayed.

Entry from: IEE23110.

Exit to: IEE23110.

IEE22110 — Display Unit Processor — Unit Status, Three

Operation: This module issues error messages, frees work area storage, and deletes the device name table.

Entry from: IEE20110, IEE23110, IEE24110.

Exit to: End of task.

IEE2303D — SMF VARY Record Handler

Operation: This module writes an SMF VARY ONLINE record.

Entry from: IEE3303D.

Exit to: IEE4203D, IEE4403D.

IEE23110 — Display Unit Processor — Unit Status, Four

Operation: This module determines the next unit address to be displayed and issues the WTO for the lines of the display.

Entry from: IEE20110.

Exit to: IEE21110, IEE22110.

Called routines: IEE21110, IEE24110.

IEE24110 — D U,,ALLOC Command Processor

Operation: This module gathers and displays the information requested on the D U,,ALLOC command. The information displayed consists of the jobname and ASID of the job(s), started task(s), or TSO user(s) to which a unit is allocated.

Entry from: IEE23110.

Exit to: IEE22110, IEE23110.

Called routines: IEECB801.

IEE3103D — VARY ONLINE/OFFLINE Processor

Operation: This module varies non-console units to either an online or an offline status.

Entry from: IEE4203D, IEE4603D.

Exit to: End of task, IEECB904.

Called routines: IEFENFFX.

IEE3203D — VARY Keyword Scan

Operation: This module scans first the primary keywords then the secondary keywords of the VARY command. It passes control according to the keyword found.

Entry from: IEE0403D.

Exit to: Variable.

IEE3303D — VARY ONLINE/OFFLINE/CONSOLE Syntax Scan

Operation: This module checks the unit field syntax, command authority, and presence of SMF in order to route control to the appropriate processor.

Entry from: IEE3603D.

Exit to: IEE4403D, IEE4203D, IEE2303D, IEECB904.

IEE3503D — DISPLAY/TRACK Router

Operation: This module routes control to a processor module based on the command operand. It processes the D T command itself.

Entry from: IEE0403D.

Exit to: Variable.

IEE3603D — VARY Command Pre-processor

Operation: This module presets the environment for the VARY ONLINE/OFFLINE/CONSOLE commands. It also keeps offline all devices without available paths.

Entry from: IEEVWAIT, IEECB904.

Exit to: IEE3303D.

IEE3703D — CANCEL/FORCE Command Handler

Operation: This module cancels or forces any background and foreground (TSO) jobs that are currently executing.

Entry from: IEE0403D.

Exit to: IEE0003D.

IEE40110 — Display PFK Processor

Operation: This module locates and displays the program function key definitions currently in effect for the operator's display console.

Entry from: IEE00110.

Exit to: Caller.

IEE4103D — Hard Copy Informational Message Module

Operation: This module issues a hard copy information message for devices that have been varied as a hard copy device.

Entry from: IEE7203D.

Exit to: IEE0003D.

IEE4203D — UCME Scanner for VARY Command

Operation: This module does syntax validation for VARY ONLINE/OFFLINE/CONSOLE commands.

Entry from: IEE4403D, IEE3303D, IEE2303D.

Exit to: IEE4903D, IEE3103D, IEE4603D, IEECB904.

IEE4303D — Processor for VARY Master Console Command

Operation: This module processes the VARY MSTCON.

Entry from: IEE3203D.

Exit to: IEE0003D.

IEE4403D — VARY Keyword Scanner

Operation: This module scans the input buffer and validates keyword values.

Entry from: IEE3303D, IEE2303D.

Exit to: IEE4203D.

IEE4603D — VARY ONLINE/OFFLINE for Console Devices

Operation: This module processes the VARY command keywords ONLINE and OFFLINE.

Entry from: IEE4203D.

Exit to: IEE3103D, IEECB904, SVC 3 (IGC003).

IEE4703D — VARY HARDCPY Operand Processor

Operation: This module begins the processing of a VARY HARDCPY Command.

Entry from: IEE3203D.

Exit to: IEE5703D, IEE7203D.

IEE4803D — Console Information Message Routine

Operation: This module uses the UCM and UCB to construct a console status message.

Entry from: IEE4903D.

Exit to: IEE7303D.

IEE4903D — Console Operand Processor

Operation: This module processes the CONSOLE Operand for the VARY command.

Entry from: IEE4203D.

Exit to: IEE4803D.

IEE5103D — ESTAE Exit Routine

Operation: This module is the SVC 34 ESTAE exit routine. It provides a storage dump in response to an ABEND situation, and it checks the validity of entries in the CSCB and CIB chains.

Entry from: ABEND STAE interface routine.

Exit to: Termination retry address in parameter list.

IEE5403D — Command Translation Routine

Operation: This module translates lower case letters, initializes the XSA, finds the beginning of a command verb and writes the command to hard copy if required.

Entry from: IEE0303D.

Exit to: IEE0403D.

IEE5503D — Stop Track and Stop Monitor Processor

Operation: This module processes the operands for both the STOPTR and STOPMN commands.

Entry from: IEE7503D, IEE0403D.

Exit to: IEE6703D, IEE0003D, IEE7103D, IEE0503D.

IEE5603D — MSGRT and CONTROL Command Message Module

Operation: This module issues error messages for CONTROL and MSGRT commands and for L=cca operands.

Entry from: Variable.

Exit to: IEE5903D, IEE0503D, or system.

IEE5703D — VARY HARDCPY, OFF Processor

Operation: This module removes the hardcopy function from a system when requested.

Entry from: IEE4703D.

Exit to: IEE0003D.

IEE5903D — Error Message Module 2 for CONTROL and MSGRT Commands

Operation: This module handles the second parts of messages for the appropriate commands and also for the L=cca operand errors.

Entry from: IEE5603D.

Exit to: IEE0003D.

IEE6303D — MSGRT Command Handler I

Operation: This module builds the route control table (RCT) for each console. This table provides default routing values.

Entry from: IEE0403D.

Exit to: IEE0503D, IEE6403D.

IEE6403D — MSGRT Command Handler II

Operation: This module formats a MSGRT command on the basis of values in the route control table.

Entry from: IEE6303D.

Exit to: IEE0003D, IEE5603D.

IEE6503D — Set Local Time

Operation: This module changes or resets the local time and date. It also updates the real time timer queue element (TQE) queue.

Entry from: IEE0603D.

Exit to: IEE0603D.

IEE6603D — Set Time-of-Day Clock Routine

Operation: This module sets a TOD clock with a time and date value that has been specified by an operator.

Entry from: IEE0603D.

Exit to: Caller.

IEE6703D — Processor for CONTROL and STOPTR Commands

Operation: This module handles the syntax checking of the K command and schedules the STOPTR command and the K command.

Entry from: IEE7503D.

Exit to: Variable.

IEE6803D — K A,nn,nn Command Handler

Operation: This module handles the processing of the K A command, which defines CRT screen areas.

Entry from: IEE6703D.

Exit to: IEE0003D, IEE5603D.

IEE6903D — K A, K T, and K M,REF Command Handler

Operation: This module handles K T, K A, K A,NONE, K M, and K M,REF commands.

Entry from: IEE6703D.

Exit to: IEE0003D, IEE5603D.

IEE70110 — Halt-End-of-Day and Switch-SMF Processor

Operation: This module processes HALT EOD and SWITCH SMF commands.

Entry from: Attached by IEEVWAIT.

Exit to: Caller.

Called routines: IEE90110 via BALR, IEECB860 (ESTAE), and IEEMB803 log task via POST.

IEE7103D — MONITOR Command Processor

Operation: This module processes the MONITOR command operands.

Entry from: IEE0403D.

Exit to: IEE0003D, IEE0503D.

IEE7203D — VARY HARDCPY Processor II

Operation: This module changes the hard copy configuration in a multiple console support (MCS) environment.

Entry from: IEE4703D.

Exit to: IEE4103D.

IEE7303D — Console Information Message Routine 2

Operation: This module constructs a message that displays the current status of a console.

Entry from: IEE4803D.

Exit to: End of task.

IEE7503D — Routing and List Operand Processor

Operation: This module processes the routing operands for DISPLAY, TRACK, CONTROL, and STOPTR commands. It also syntax checks the secondary operands (LIST, ALL, and name) for the DISPLAY and TRACK commands.

Entry from: IEE3503D, IEE0403D, JES2 routine.

Exit to: IEE0803D, IEE5503D, IEE6703D, IEE5603D, IEE0503D, IEE9403D, JES2 routine.

IEE7703D — CONTROL V Command Handler

Operation: This module processes CONTROL V commands.

Entry from: IEE6703D.

Exit to: IEE0003D, IEE5603D.

IEE7803D — CONTROL C,D Command Handler

Operation: This module processes the CONTROL C,D command for cancelling an inline MLWTO.

Entry from: IEE6703D.

Exit to: IEE0003D, IEE5603D.

IEE7903D — CONTROL M Command Handler

Operation: This module processes CONTROL M commands for activating or deactivating the action message retention facility and changing the WQE buffer limits.

Entry from: IEE6703D.

Exit to: IEE0503D, IEE5603D.

IEE8B03D — CONTROL Q Command Processor

Operation: This module processes the CONTROL Q command for re-routing messages.

Entry from: IEE6703D.

Exit to: IEE0003D.

IEE8103D — CONTROL C,A/E/I Command Handler

Operation: This module deletes retained action messages in response to a CONTROL C,A, CONTROL C,E, or CONTROL C,I command.

Entry from: IEE6703D.

Exit to: IEE5603D.

Called routines: IEAVQ700.

IEE8603D — SETDMN Command Processor

Operation: This module passes new domain description values to the system resources manager (SRM).

Entry from: IEE0403D.

Exit to: IEE0003D.

IEE90110 — HALT/SWITCH Message Module

Operation: This module assembles and edits messages for HALT and SWITCH commands, and issues a WTO.

Entry from: IEE70110.

Exit to: IEE70110.

IEE9403D — MSS Preprocessor

Operation: If MSS is active this module prepares MSS commands for interfacing to MSS routines, then posts the MSS routines.

Entry from: IEE0403D, IEE1403D, IEE3203D, IEE7503D.

Exit to: Caller.

IEFAB4A0 — Common Unallocation Control

Operation: This module controls the processing necessary to unallocate a request that has been allocated by means of either JCL or a dynamic allocation. The functions provided are:

- Disposition processing

- Disconnecting of private catalogs
- Data set release (DEQ)
- Unit unallocation
- Volume release (DEQ)
- Release of DSAB/TIOT entries

Entry from: IEFAB4DE, IEFAB477, IEFAB490, IEFAB492, IEFBB414, IEFBB416, IEFDB4A1, IEFDB413, IEFDB418.

Exit to: Caller.

Called routines: IEEAB401, IEFAB4A2, IEFAB4A4, IEFAB4A6, IEFAB4A8, IEFAB4FC, IEFAB4F4.

IEFAB4A2 — Disposition Processing Control

Operation: This module controls the processing required to dispose of data sets as indicated in the input requests.

Entry from: IEFAB4A0.

Exit to: Caller.

Called routines: IEFAB4A3, IEFAB4B0, IEFAB4B2, IEFAB4SF.

IEFAB4A3 — Special Scratch/Catalog Processing

Operation: This module builds a dummy ETIOT needed by the scratch function of data management, when no DSABs and TIOTs exist.

This module also interfaces with IEFAB4F5 to allocate and open step catalogs, if requested, and with IEFAB4F4 to unallocate all private catalogs when the disposition processing has completed, if the allocation of step catalogs was requested.

Entry from: IEFAB4A2.

Exit to: Caller.

Called routines: IEFAB4FC, IEFAB428, IEFAB4F4, IEFAB4F5.

IEFAB4A4 — Unit Unallocation Control

Operation: This module controls the processing associated with unallocating devices for each request. The functions performed are:

- Releasing units no longer needed
- Rewinding tape volumes
- Deleting outstanding mount messages
- Identifying volumes to be unloaded
- Posting waiting allocations

Entry from: IEFAB4A0, IEFAB4DE.

Exit to: Caller.

Called routines: IEFAB4FA, IEFAB4F7, IEFAB493, IEFHB410.

IEFAB4A6 — Data Set Release

Operation: This module releases (dequeues) data sets that are no longer needed by the job.

Entry from: IEFAB4A0, IEFBB401.

Exit to: Caller.

Called routines: IEFAB4F7.

IEFAB4A8 — Volume Release

Operation: This module is responsible for releasing volumes that were reserved for the job step by allocation and data management (OPEN/CLOSE/EOV).

Entry from: IEFAB4A0.

Exit to: Caller.

Called routines: None.

IEFAB4B0 — Disposition Message Routine

Operation: This module issues disposition messages to the system output data set and to the system operator.

Entry from: IEFAB4A2.

Exit to: Caller.

Called routines: IEFAB4FD.

IEFAB4B2 — Alternate Disposition Message Routine

Operation: This module issues disposition messages to the system output data set and/or to the system operator when no storage is available for a volume list.

Entry from: IEFAB4A2.

Exit to: Caller.

Called routines: IEFAB4FD.

IEFAB4DC — Data Set Reservation/Release

Operation: This module can perform the following functions:

- Updates the data set enqueue table to add or change an entry
- Dequeues the dsname
- Updates the data set enqueue table to delete an entry

Entry from: IEFAB459, IEFDB411, IEFDB413, IEFDB418.

Exit to: Caller.

Called routines: IEFAB4F7.

IEFAB4DD — Job/Step Unallocation ESTAE Exit

Operation: In the event of an abnormal termination, this module calls step and/or job unallocation, if needed, and if SWA records have been read successfully. This module also requests a dump, if necessary.

Entry from: RTM (recovery termination management).

Exit to: Caller.

Called routines: IEEAB400, IEEAB401, IEFBB414, IEFBB416.

IEFAB4DE — Common Unallocation ESTAE Exit Routine

Operation: If an abnormal termination occurs in a module that had requested a retry, IEFAB4DE puts the address of the allocation ESTAE retry block (AERB) into the SDWACOMU field of the SDWA and returns to the caller. Otherwise, this module attempts to perform any common unallocation functions that had been requested but not yet attempted at the time of an abnormal termination.

Entry from: IEFAB4ED.

Exit to: Caller.

Called routines: IEFAB4A0, IEFAB4A4, IEFAB4FA.

IEFAB4EA — Housekeeping ESTAE Exit Routine

Operation: This module reestablishes control block pointers and frees resources for JFCB housekeeping control (IEFAB451), allocate catalog control (IEFAB4F5), or unallocate catalog control (IEFAB4F4), if an ESTAE is scheduled in any of those routines as the result of an ABEND.

Entry from: IEFAB4ED.

Exit to: Caller.

Called routines: IEFAB4F4.

IEFAB4EB — PDI Read and Chain

Operation: This module reads all the PDIBs (passed data set information blocks) that comprise the PDI (passed data set information) and chains the PDIBs together.

Entry from: IEFAB455, IEFAB459, IEFBB416.

Exit to: Caller.

Called routines: IEFAB4F7.

IEFAB4EC — Group Lock/Unlock Interface

Operation: This module obtains locks for the groups represented by the list of UCBs passed to it. It performs this serialization via the allocation queue manager.

Entry from: IEFBB416.

Exit to: Caller.

Called routines: IEFAB4FA, IEFAB4UV.

IEFAB4ED — Allocation Common ESTAE Exit

Operation: This module is the common ESTAE exit for all of allocation. It performs general recovery processing and then routes control to other exit routines for more specific recovery. If a specific recovery routine requests a retry, this module returns to RTM requesting a retry. Otherwise, after all of the recovery routines have completed processing, this module returns to RTM requesting percolation.

Entry from: RTM (recovery termination management).

Exit to: Caller.

Called routines: IEEAB401, IEFAB4DD, IEFAB4DE, IEFAB4EA, IEFAB4E2, IEFAB4E4, IEFAB4E7, IEFAB4E8, IEFDB402.

IEFAB4EE — Allocation Message Routine

Operation: This module builds the allocation messages for the requests on the input SIOT chain.

Entry from: IEFAB490, IEFBB414.

Exit to: Caller.

Called routines: IEFAB4FD, IEFAB4F3.

IEFAB4EF — PCCB Routine

Operation: This module handles all requests for the creation, modification, and deletion of PCCB(s) and their chain structure.

Entry from: IEFAB4F4, IEFAB4F5, IEFAB452.

Exit to: Caller.

Called routines: None.

IEFAB4EG — Allocation Resource Manager ESTAE Exit Routine

Operation: If the allocation resource manager (IEFAB4F5) abnormally terminates and had requested a retry, IEFAB4EG puts the address of the allocation ESTAE retry block (AERB) into the SDWACOMU field of the SDWA.

Entry from: IEFAB4ED.

Exit to: Caller.

Called routines: None.

IEFAB4E0 — Test If Device Is Ready

Operation: This module tests to determine whether a device is physically ready by issuing a NOP EXCP to the device.

Entry from: IEFAB473, IEFAB49C, IEFAB494, IEFAB495.

Exit to: Caller.

Called routines: None.

IEFAB4E1 — VM&V FRR Exit Routine

Operation: This module receives control if an ABEND occurred while an MVCA was being dechained. It tries again to take the MVCA off the chain. If successful, it sets a bit to indicate that it should free MVCA storage.

Entry from: RTM (recovery termination management).

Exit to: Caller.

Called routines: None.

IEFAB4E2 — VM&V ESTAE Exit Routine

Operation: If an abnormal termination occurs in a module that had requested a retry, IEFAB4E2 puts the address of the allocation ESTAE retry block (AERB) into the SDWACOMU field of the SDWA and returns to the caller. Otherwise, this module releases mount control blocks that still exist for this allocation.

Entry from: IEFAB4ED.

Exit to: Caller.

Called routines: IEFAB49A, IEFAB498.

IEFAB4E3 — Job/Step Allocation Retry Routine

Operation: This retry routine receives the initiator's registers on entry. It sets a return code of 4 and branches on register 14, thereby simulating an error return to the initiator.

Entry from: RTM (recovery termination management).

Exit to: IEFSD162.

Called routines: None.

IEFAB4E4 — Job/Step Allocation ESTAE Exit

Operation: This module sets bits in the LCT to indicate that an error has occurred. It then prepares for retry by storing the initiator's registers in the SDWA. These registers will be received by retry routine IEFAB4E3.

Entry from: IEFAB4ED.

Exit to: Caller.

Called routines: IEEAB400, IEEAB401, IEFAB4F7.

IEFAB4E5 — Allocation Resource Manager

Operation: This module attempts to clean up system resources owned by allocation in the event of an abnormal memory termination. It does the following:

- Releases device groups held by the terminating address space
- Frees cross memory communication areas
- Posts other allocations waiting for devices
- Unallocates UCBs allocated to this address space

Entry from: RTM (recovery termination management).

Exit to: Caller.

Called routines: IEFAB4FA, IEFAB498, IEFHB410.

IEFAB4E6 — Allocation Recovery Routine

Operation: This module handles abnormal terminations that occur when the allocation address space is being initialized or updated or when the UCB field UCBUSER is being updated. It also checks the counts in the UCB and DALT to ensure that they are correct.

Entry from: RTM.

Exit to: Caller.

Called routines: IEFHB410 by means of a program call (PC).

IEFAB4E7 — Group Lock/Unlock ESTAE Exit

Operation: This module frees resources held by LOCK/UNLOCK processing (IEFAB4EC) if an ABEND occurs while IEFAB4EC's ESTAE is in effect.

Entry from: IEFAB4ED.

Exit to: Caller.

Called routines: IEFAB4FA.

IEFAB4E8 — Common Allocation ESTAE Exit Routine

Operation: If an abnormal termination occurs in a module that had requested a retry, IEFAB4E8 puts the address of the allocation ESTAE retry block (AERB) into the SDWACOMU field of the SDWA and returns to the caller. Otherwise, this module performs cleanup processing if an abnormal termination has occurred during common allocation processing.

Entry from: IEFAB4ED.

Exit to: Caller.

Called routines: IEEAB401, IEFAB4FA, IEFAB49A, IEFAB498.

IEFAB4E9 — Dynamic Allocation Set DSABDCBM Mask Bits Subroutine

Operation: For each filled-in DCB field in the JFCB, this module sets on the corresponding bit in DSABDCBM.

Entry from: IEFAB428, IEFDB411.

Exit to: Caller.

Called routines: None.

IEFAB4FA — Allocation Queue Manager

Operation: This module serializes access to UCB (device) groups by device allocation and manages the WAIT-FOR-DEVICE queue.

Entry from: IEFAB4A4, IEFAB4DE, IEFAB4E5, IEFAB4E8, IEFAB471, IEFAB472, IEFAB485, IEFAB486, IEFAB490, IEFAB491, IEFAB492.

Exit to: Caller.

Called routines: IEA0PT01 (system post).

IEFAB4FC — TIOT Manager Control Routine

Operation: This module is responsible for obtaining and maintaining the TIOT and the DSAB queue descriptor block (QDB). The functions of this routine are:

- Create the TIOT and DSAB QDB
- Initialize TIOT and DSAB QDB.
- Allocate a TIOT DD entry
- Release a TIOT DD entry
- Update the TIOT and DSAB QDB
- Unallocate the DSAB/TIOT(ETIOT) entry
- Concatenate the DSAB/TIOT DD entry
- Free the TIOT and QDB

Entry from: IEESB601, IEFAB4A0, IEFAB4A3, IEFAB428, IEFBB401, IEFBB404, IEFBB410, IEFDB413, IEFDB450.

Exit to: Caller.

Called routines: None.

IEFAB4FD — System Message Interface Routine

Operation: This module converts the input parameter list into a WTO parameter list, which it passes to IEEAB400. IEEAB400 buffers any WTO messages (routing code 11) and issues the WTO macro instruction if any other routing codes are specified.

Entry from: IEFAB4B0, IEFAB4B2, IEFAB4EE, IEFAB490, IEFBB401, IEFBB402, IEFBB404, IEFBB410.

Exit to: Caller.

Called routines: IEEAB400.

IEFAB4FE — SWA Reader Routine

Operation: This module establishes addressability to SIOTs and their related JFCBs and JFCBXs.

Entry from: IEFBB401, IEFBB410, IEFDB400.

Exit to: Caller.

Called routines: IEFAB4F7.

IEFAB4F0 — Generalized Conditional ENQ/DEQ Routine

Operation: This module does the following:

- Enqueues conditionally (either shared or exclusive) on the input volume serial number
- Dequeues conditionally on the input volume serial number

Entry from: IEFAB436, IEFAB497, IEFAB492.

Exit to: Caller.

Called routines: None.

IEFAB4F1 — Test Volume ENQ

Operation: This module determines if the input volume serial number can be shared by this allocation (even though it is already being used by this job step through a prior entry to allocation).

Entry from: IEFAB421.

Exit to: Caller.

Called routines: None.

IEFAB4F2 — Update Algorithm Tables

Operation: This module helps maintain the cover/reduce algorithm interface tables. Its functions are:

- Change a request to device-required.
- Indicate a request was just allocated.

Entry from: IEFAB432, IEFAB434, IEFAB442.

Exit to: Caller.

Called routines: IEFAB481.

IEFAB4F3 — Message Compression Routine

Operation: This module eliminates consecutive and trailing blanks from the text received in the input buffer. The compressed text and the new text length are returned to the caller.

Entry from: IEFAB4EE, IEFAB421, IEFAB48A, IEFAB487, IEFAB488, IEFAB490, IEFBB401, IEFBB404, IEFBB410.

Exit to: Caller.

Called routines: None.

IEFAB4F4 — Catalog Unallocation Control

Operation: This module calls other routines to perform the following functions, as requested by the caller:

- Close private catalogs
- Unallocate private catalogs
- Release private catalog control blocks (PCCBs)

Entry from: IEFAB4A0, IEFAB4EA, IEFAB4F5, IEFAB451.

Exit to: Caller.

Called routines: IDACAT12, IEFAB4EF, IEFDB400 (via SVC99).

IEFAB4F5 — Allocate Catalog Control

Operation: This module can:

- Allocate a catalog.
- Cause an ACB to be created and opened.
- Cause a PCCB to be created and chained or modified.

Entry from: IEFAB469, IEFICATL (an initiator module), IGG0CLCA (a catalog management module), IGG0CLC9 (a catalog management module).

Exit to: Caller.

Called routines: IDACAT11, IEFAB4EA, IEFAB4EF, IEFAB4F4.

IEFAB4F6 — GSPACE/FSPACE Service Routines

Operation: This module obtains storage and subdivides them as storage requests are made. Requests for release of the 4K blocks and the obtained sections of them are also serviced.

Entry from: Most allocation/unallocation modules, that is, modules with names beginning with IEFAB, IEFBB, or IEFDB.

Exit to: Caller.

Called routines: None.

IEFAB4F7 — SWA Manager Interface

Operation: This module invokes the SWA Manager to assign, read, write, or delete blocks of the Scheduler Work Area and returns the necessary parameters to the caller.

Entry from: IEFAB4A4, IEFAB4A6, IEFAB4DC, IEFAB4EB, IEFAB4FE, IEFAB4SF, IEFAB427, IEFAB452, IEFAB453, IEFAB455, IEFAB457, IEFAB458, IEFAB459, IEFAB461, IEFAB464, IEFAB466, IEFBB401, IEFBB402, IEFBB410, IEFBB416, IEFDB4A1, IEFDB400, IEFDB410, IEFDB413, IEFDB414, IEFDB418, IEFDB450.

Exit to: Caller.

Called routines: None.

IEFAB4F8 — Direct Access Label Read

Operation: This module reads direct access volume labels.

Entry from: IEFAB473, IEFAB49B.

Exit to: Caller.

Called routines: None.

IEFAB4F9 — Tape Label Read

Operation: This module reads tape volume labels.

Entry from: IEFAB473.

Exit to: Caller.

Called routines: None.

IEFAB4GB — Modify Device Online/Offline Status

Operation: This module designates the device status in the UCB as either offline or pending offline, depending on the input function code.

Entry from: IEFAB429, IECVRDIO, IECVRSTI, IECVFDEV, IECVRRSV.

Exit to: Caller.

Called routines: None.

IEFAB4M4 — Volume Mount & Verify Messages

Operation: This module contains the text used to construct the messages issued by volume mount and verify routines.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFAB4M5 — WTO Message Module For AVR and Common Allocation

Operation: This module contains the list forms of WTO and WTOR messages issued by common allocation, automatic volume recognition (AVR), and data set reservation/release.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFAB4M6 — Disposition Messages

Operation: This module contains the text used to construct the disposition messages issued by common unallocation.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFAB4M7 — Allocation Messages

Operation: This module contains the allocation messages that are issued to the system output data set and, for unit record devices, to the console.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFAB4M9 — Recovery Allocation Messages

Operation: This module contains the text used to construct recovery allocation operator messages.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFAB4SF — Spool File Allocation

Operation: This module interfaces with the job entry subsystem to allocate or unallocate SYSIN and SYSOUT data sets. It is also called to segment a SYSOUT data set.

Entry from: IEFAB4A2, IEFAB427, SETPRT SVC.

Called routines: IEFAB4F7, subsystems.

IEFAB4UV — Unit Verification

Operation: This routine is invoked by system routines that require information or validity checks based on the contents of the eligible devices table (EDT).

Entry from: IAPINMD, ICBME.

Exit to: Caller.

Called routines: None.

IEFAB421 — Common Allocation Control

Operation: This module is called to allocate one or more data set requests. It is responsible for controlling the following functions:

- Process pending offlines/unloads
- Initialize work area (ALCWA)
- Allocate DUMMY requests
- Allocate VIO requests
- Allocate subsystem requests
- Determine device requirements
- Allocate to fixed DA devices
- Allocate TP requests
- Volume reservation
- Allocate by device type
- Recovery allocation
- Dequeue from allocation resource
- Cleanup processing

Entry from: IEFAB490, IEFBB404, IEFDB413.

Exit to: Caller.

Called routines: IEEAB401, IEFAB4F3, IEFAB422, IEFAB423, IEFAB425, IEFAB427, IEFAB428, IEFAB429, IEFAB430, IEFAB431, IEFAB471, IEFAB485, IEFAB490, IEFAB491.

IEFAB422 — JES3 Interface

Operation: This module interfaces with the JES3 subsystem and then updates allocation tables based on whether or not JES3 has made device selections.

Entry from: IEFAB421.

Exit to: Caller.

Called routines: None.

IEFAB423 — Determine Device Requirements

Operation: The function of this module is to determine device eligibility for each individual request and for the step. Device requirements are represented by the volunit table and device eligibility is represented by the eligible device table (EDT) and eligible device list (EDL). This module builds the volunit table and calls IEFAB424 to build the EDL.

Entry from: IEFAB421.

Exit to: Caller.

Called routines: IEFAB424, IEFAB426.

IEFAB424 — Build Eligible Device List (EDL)

Operation: This module is responsible for construction of an eligible device list (EDL) for each request.

Entry from: IEFAB423.

Exit to: Caller.

Called routines: IEFAB438.

IEFAB425 — Process TP Requests

Operation: This module controls the processing necessary to allocate teleprocessing requests.

Entry from: IEFAB421.

Exit to: Caller.

Called routines: IEFAB434.

IEFAB426 — Volunit Affinity Processing

Operation: This module updates volunit table entries to reflect volume affinity. It ensures that every member of a volume affinity has the same unit identification.

Entry from: IEFAB423.

Exit to: Caller.

Called routines: None.

IEFAB427 — Allocate Subsystem Data Sets

Operation: This module controls allocation of subsystem data sets. The major functions performed are:

- Find subsystem requests
- Sort subsystem requests
- Issue the IEFSSREQ macro instruction
- Create DSAB/TIOT entries

Entry from: IEFAB421.

Exit to: Caller.

Called routines: IEFAB4F7, IEFAB4SF, subsystems, IEFAB428.

IEFAB428 — Build/Update/Rebuild Expandable TIOT

Operation: This module creates, updates, or rebuilds the expandable TIOT (DSAB/TIOT DD entry).

Entry from: IEFAB4A3, IEFAB421, IEFAB427, IEFAB431, IEFAB434, IEFAB442.

Exit to: Caller.

Called routines: IEFAB4E9, IEFAB4FC.

IEFAB429 — Process Pending Offlines/Unloads

Operation: This module performs offline/unload processing on units or consoles, after the device is unallocated.

Entry from: IEFAB421.

Exit to: IEFAB421.

Called routines: IEFAB4GB, IEFAB493, IEFENFFX.

IEFAB430 — Fixed Device (Main Path) Control

Operation: This module controls the processing necessary to allocate requests to direct access volumes that are permanently resident or reserved.

Entry from: IEFAB421.

Exit to: Caller.

Called routines: IEFAB433, IEFAB436.

IEFAB431 — Allocate VIO Data Sets

Operation: This module controls the allocation of VIO data set requests.

Entry from: IEFAB421.

Exit to: Caller.

Called routines: IEFAB428.

IEFAB432 — Affinity Processor

Operation: This module processes any affinity requests to the volume or unit just allocated. Processing includes ensuring that the affinity requests are valid and, possibly, allocating the requests.

Entry from: IEFAB434.

Exit to: Caller.

Called routines: IEFAB4F2, IEFAB434, IEFAB441.

IEFAB433 — Specific Volume Allocation Control

Operation: This module controls the processing necessary to allocate specific volume requests when the volume is mounted. A specific request is one that requires a particular volume — for example, a volume serial number was specified, the data set was passed, the data set is cataloged.

Entry from: IEFAB430, IEFAB475, IEFAB479, IEFAB485.

Exit to: Caller.

Called routines: IEFAB434, IEFAB442.

IEFAB434 — Allocate Request to Unit

Operation: This module controls the processing necessary to allocate a request to the selected device. Functions include:

- Updating the UCB of the selected device (done by IEFAB435)
- Creating or updating a DSAB and TIOT entry (done by IEFAB428)
- Acquiring space on the volume (interfacing with DADSM)
- Unallocating a unit and decreasing the DALT use count (done by IEFHB410), if a DADSM allocate error occurs
- Processing requests that specify affinity to the request being handled (done by IEFAB432)

Entry from: IEFAB425, IEFAB432, IEFAB433, IEFAB436, IEFAB441, IEFAB478, IEFAB479, IEFAB489.

Exit to: Caller.

Called routines: IEFAB4F2, IEFAB428, IEFAB432, IEFAB435, IEFHB410.

IEFAB435 — Update UCB Routine

Operation: This module updates the UCB to reflect allocation of the current request to it. It also interfaces with VM&V control to unload volumes mounted on the needed unit.

Entry from: IEFAB434.

Exit to: Caller.

Called routines: IEFAB4E0, IEFAB441, IEFAB493, IEFHB410.

IEFAB436 — Nonspecific Volume Allocation Control

Operation: This module controls the processing necessary to allocate nonspecific, nonprivate requests to public and storage volumes. It is responsible for building a list of devices that can satisfy the request, interfacing with the system resources manager to select the device to be allocated, and interfacing with IEFAB434 to allocate the request.

Entry from: IEFAB430, IEFAB475, IEFAB485.

Exit to: Caller.

Called routines: IEFAB4F0, IEFAB434, IEFAB440, IEFAB442.

IEFAB438 — Update DDR Count Routine

Operation: This module retrieves the DDR count from the CSD. The count is used by module IEFAB424 to ensure that allocation is synchronized with DDR while the eligible device list (EDL) is being built.

Entry from: IEFAB424.

Exit to: Caller.

Called routines: None.

IEFAB440 — Build Allocated UCB List

Operation: This module builds a list of allocated UCBs (ALCUCBLT) which is used in the interface with the system resources manager.

Entry from: IEFAB436, IEFAB478, IEFAB489, IEFAB490.

Exit to: Caller.

Called routines: None.

IEFAB441 — Volume Validity Checker

Operation: This module determines if the input volume is currently mounted. If it is, this module checks to see if the volume can be used by the current request.

Entry from: IEFAB432, IEFAB435, IEFAB486.

Exit to: Caller.

Called routines: IEFAB434, IEFAB442, IEFAB493.

IEFAB442 — Affinity Remover

Operation: This module cancels unit affinity between the input request and other requests, when the volume being allocated to the input request is permanently resident or reserved.

Entry from: IEFAB433, IEFAB436, IEFAB441, IEFAB479.

Exit to: Caller.

Called routines: IEFAB4F2, IEFAB428.

IEFAB445 — Device Allocation Defaults Module

Operation: This module contains space and unit name defaults used by device allocation.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFAB451 — JFCB Housekeeping Control

Operation: This module is responsible for retrieving the information necessary for allocation.

Entry from: IEFBB404, IEFDB413.

Exit to: Caller.

Called routines: IEFAB4EA, IEFAB4F4, IEFAB452.

IEFAB452 — DD Processing Control

Operation: This module selects and initiates the processing of:

- STEPCAT requests
- Other data set requests

Entry from: IEFAB451.

Exit to: Caller.

Called routines: IEFAB4EF, IEFAB4F7, IEFAB453, IEFAB454.

IEFAB453 — DD Preparation

Operation: This module does the following processing:

- Initializes the pointer to the JFCB for the requests
- Processes data set requests that specified QNAME, TERM=TS, DUMMY, DSN=NULLFILE, or PGM=*, stepname.ddname or PGM=*, procstepname.ddname
- Processes subsystem and VIO data sets
- Retrieves unit information

Entry from: IEFAB452.

Exit to: Caller.

Called routines: IEFAB4F7, IEFAB470.

IEFAB454 — DD Function Control

Operation: This module determines what additional information is required for a DD. It then constructs a parameter list and routes control to the appropriate routine(s) to obtain that information.

Entry from: IEFAB452.

Exit to: Caller.

Called routines: IEFAB456, IEFAB457, IEFAB458, IEFAB459, IEFAB469.

IEFAB455 — Passed Data Set Information Scan

Operation: This module scans the PDI (passed data set information) for the data set indicated. If the data set is found, it returns a pointer (SVAB) to the SIOT that passed the data set.

Entry from: IEFAB469.

Exit to: Caller.

Called routines: IEFAB4EB, IEFAB4F7.

IEFAB456 — Data Set Name Resolution

Operation: This module resolves data set name information and provides volume and unit information for GDG requests.

Entry from: IEFAB454.

Exit to: Caller.

Called routines: IEFAB461, IEFAB466, IEFAB469.

IEFAB457 — Volume/Unit Resolution

Operation: This module receives control for every request in order to resolve volume and unit information for subsequent allocation processing.

Entry from: IEFAB454.

Exit to: Caller.

Called routines: IEFAB4F7, IEFAB463, IEFAB464, IEFAB466, IEFAB469.

IEFAB458 — DCB Resolution

Operation: This module establishes DCB control information in a JFCB.

Entry from: IEFAB454.

Exit to: Caller.

Called routines: IEFAB4F7, IEFAB469.

IEFAB459 — Disposition Processing

Operation: This module does the following:

- Completes data set disposition information in the SIOT
- Updates the PDI (passed data set information), if necessary
- Marks tape data set requests as private, if necessary
- Retrieves catalog information, if necessary

Entry from: IEFAB454.

Exit to: Caller.

Called routines: IEFAB4DC, IEFAB4EB, IEFAB469.

IEFAB461 — GDG Single Processing

Operation: This module converts a GDG (generation data group) data set request with a relative data set name (generation number) to a fully qualified data set name. It obtains volume and unit information for a GDG single request of an existing data set.

Entry from: IEFAB456.

Exit to: Caller.

Called routines: IEFAB4F7, IEFAB469.

IEFAB463 — Multiple Device Type Determination

Operation: This module scans the catalog return information(CRI) to:

- Determine if the data set is a multi-device type data set and, if so, to return the number of different device types minus one

- Given the number of device types, find and return the respective device type and a pointer to the volume list entry for that device type

Entry from: IEFAB457, IEFAB464.

Exit to: Caller.

Called routines: None.

IEFAB464 — Volume Unit Table Completion

Operation: This module copies appropriate volume and unit information from the source determined by previous routines.

Entry from: IEFAB457.

Exit to: Caller.

Called routines: IEFAB4F7, IEFAB463, IEFAB470.

IEFAB466 — Table Creation

Operation: The module creates SIOTs/JFCBs and initializes them.

Entry from: IEFAB456, IEFAB457, IEFAB469.

Exit to: Caller.

Called routines: IEFAB4FD, IEFAB4F7.

IEFAB469 — JLOCATE

Operation: JLOCATE is a housekeeping service routine that provides data set name information for GDG single requests and necessary volume and unit information for a given data set name.

Entry from: IEFAB454, IEFAB456, IEFAB457, IEFAB458, IEFAB459, IEFAB461.

Exit to: Caller.

Called routines: IEFAB4EF, IEFAB4F5, IEFAB466.

IEFAB470 — Unit Name Conversion

Operation: This module extracts necessary EDT or UCB device information from the current SIOT.

Entry from: IEFAB453, IEFAB464.

Exit to: Caller.

Called routines: None.

IEFAB471 — Generic Allocation Control

Operation: This module controls the allocation of requests that could not be handled by fixed device allocation. (See IEFAB430.) It handles requests to any mounted volumes and/or to any online and unallocated units. It processes one generic device type at a time. The order of processing is determined by the installation device precedence list.

Entry from: IEFAB421.

Exit to: Caller.

Called routines: IEFAB4FA, IEFAB472, IEFAB473, IEFAB475.

IEFAB472 — Build Tables for Generic Processing

Operation: This module builds the tables needed for generic allocation processing. The tables built are:

- Allocation queue manager request block
- Cover/reduce algorithm interface tables
- Request identification mask table
- Workarea for manipulating group masks

Entry from: IEFAB471.

Exit to: Caller.

Called routines: IEFAB4FA, IEFAB481.

IEFAB473 — AVR (Automatic Volume Recognition) Control

Operation: This module reads the volume label of any volumes mounted on units whose UCBs are passed as input.

Entry from: IEFAB471.

Exit to: Caller.

Called routines: IEFAB4E0, IEFAB4F8, IEFAB4F9, IEFAB493, IEFXVNSL.

IEFAB474 — Process Multi-Unit/Multi-Generic Requests

Operation: This module determines if a multi-unit (multi-generic) request processed by the algorithm is assigned to a single generic device type. If not, this module attempts to find a single generic device type that can satisfy the request.

Entry from: IEFAB476, IEFAB486.

Exit to: Caller.

Called routines: IEFAB481.

IEFAB475 — Allocate within a Generic

Operation: This module attempts to allocate as many requests as possible to the chosen generic device type. It determines what types of allocation are required and then calls the necessary allocation routines.

Entry from: IEFAB471.

Exit to: Caller.

Called routines: IEFAB433, IEFAB436, IEFAB476, IEFAB479.

IEFAB476 — Allocate via the Algorithm

Operation: This module processes requests when a choice of units exists, that is when a specific unit is not requested; the request cannot be satisfied by allocating it to a mounted volume; or the request cannot be allocated to a permanently resident or reserved volume. IEFAB476 calls the cover/reduce algorithm (IEFAB480) to select the device groups from which to allocate.

Entry from: IEFAB475, IEFAB485.

Exit to: Caller.

Called routines: IEFAB474, IEFAB477, IEFAB478, IEFAB480.

IEFAB477 — Unallocate Requests to be Rearranged

Operation: This module unallocates requests that the algorithm indicates must be rearranged.

Entry from: IEFAB476, IEFAB485, IEFAB486.

Exit to: Caller.

Called routines: IEFAB4A0, IEFAB493, IEFHB410.

IEFAB478 — Allocate From Groups Picked by Algorithm

Operation: This module allocates requests to available (that is, online, unallocated) devices within the device groups selected by the algorithm.

Entry from: IEFAB476, IEFAB486, IEFAB491.

Exit to: Caller.

Called routines: IEFAB434, IEFAB440, ICBME.

IEFAB479 — Demand Allocation

Operation: This module allocates requests that specify a specific unit address (for example, unit=190) within the generic device type being processed by generic allocation.

Entry from: IEFAB475, IEFAB485, IEFAB491.

Exit to: Caller.

Called routines: IEFAB4F0, IEFAB433, IEFAB434, IEFAB442.

IEFAB48A — Process Offlines/Allocated Units

Operation: This module controls the processing necessary:

- To determine whether a request can be satisfied by offline and/or allocated units
- To inform the operator of the possible options

Entry from: IEFAB487.

Exit to: Caller.

Called routines: IEFAB4F3, IEFAB488, IEFAB489.

IEFAB480 — Cover/Reduce Algorithm

Operation: This module uses the hungarian algorithm to determine which group or groups of devices should be used to satisfy a generic allocation request.

Entry from: IEFAB476, IEFAB486, IEFAB488, IEFAB489, IEFAB491.

Exit to: Caller.

Called routines: None.

IEFAB481 — Eliminate Ineligible Groups and Generics

Operation: This module marks all generic device types, except the one selected for the current request, as ineligible in the EDL. It also marks group list entries in the algorithm tables as ineligible, when the groups are not in the selected generic device type.

Entry from: IEFAB4F2, IEFAB472, IEFAB485, IEFAB486.

Exit to: Caller.

Called routines: None.

IEFAB485 — Recovery Allocation

Operation: This module handles allocation requests that could not be allocated by either Fixed Device Control or Generic Allocation. Offline and/or allocated units can be considered during these allocations, if necessary and if allowed by the caller.

Entry from: IEFAB421.

Exit to: Caller.

Called routines: IEFAB4FA, IEFAB433, IEFAB436, IEFAB476, IEFAB477, IEFAB479, IEFAB481, IEFAB486, IEFAB49C.

IEFAB486 — Offline/Allocated Device Allocation

Operation: This module handles allocation processing when it is necessary to consider offline and/or allocated devices.

Entry from: IEFAB485.

Exit to: Caller.

Called routines: IEFAB4FA, IEFAB441, IEFAB474, IEFAB477, IEFAB478, IEFAB480, IEFAB481, IEFAB487, IEFAB493.

IEFAB487 — Allocation Recovery Interface With Operator

Operation: This module controls the processing necessary to inform the system operator that this allocation cannot proceed without waiting for devices to be unallocated by other jobs and/or bringing needed devices online.

Entry from: IEFAB486.

Exit to: Caller.

Called routines: IEFAB4F3, IEFAB48A, IEFAB488.

IEFAB488 — Recovery Reply Options Processor

Operation: This module builds and issues the appropriate reply options message and processes the operator's reply.

Entry from: IEFAB48A, IEFAB487.

Exit to: Caller.

Called routines: IEFAB4F3, IEFAB480, IEFAB489, IEFENFFX.

IEFAB489 — Recovery Allocation of Online Devices

Operation: This module determines, for each group of UCBs in the system, whether the number of online units in the group has increased as a result of the system operator's responses to the reply options message. If it has, the routine will attempt to allocate requests to the units in the group.

Entry from: IEFAB48A, IEFAB488.

Exit to: Caller.

Called routines: IEFAB434, IEFAB440, IEFAB480, ICBME.

IEFAB49A — VM&V DOMR & Cleanup Routine

Operation: This module deletes mount messages, if necessary, and release used by this allocation when all mount messages have been deleted.

Entry from: IEFAB4E2, IEFAB4E8, IEFAB492, IEFAB496.

Exit to: Caller.

Called routines: IEFAB498.

IEFAB49B — Device End POST Handler

Operation: This module verifies, if necessary, that:

- A device is now ready (that is, a volume has been mounted); or,
- The volume mounted on the device is acceptable in response to a mount message issued by mount control or a mount request to the 3850 mass storage system.

Entry from: IEFAB496.

Exit to: Caller.

Called routines: IEFAB4F3, IEFAB4F8, IEFAB493.

IEFAB49C — Non-Allocation Unload Interface

Operation: This module interfaces with volume mount and verify (VM&V) control (IEFAB493) to have volumes unloaded and/or rewound for non-allocation callers. Before calling IEFAB493, this routine verifies the UCB pointers in each VM&V request block (RB).

Entry from: IATLVVR, IATSICA, IEEMB813, IEEVCPRL, or IEEVPTH.

Exit to: Caller.

Called routines: IEFAB493.

IEFAB490 — Common Allocation Cleanup

Operation: This module performs cleanup functions when allocation has completed or when a terminating allocation error has been detected.

Entry from: IEFAB421.

Exit to: Caller.

Called routines: IEFAB4A0, IEFAB4EE, IEFAB4FA, IEFAB4FD, IEFAB4F3, IEFAB421, IEFAB492.

IEFAB491 — Wait Holding Resources

Operation: This module waits for other job steps to unallocate devices in the device groups from which this allocation needs additional units. When posted because of an unallocation, this routine tries to satisfy the remaining requests.

Entry from: IEFAB421.

Exit to: Caller.

Called routines: IEFAB4FA, IEFAB478, IEFAB479, IEFAB480, IEFAB493.

IEFAB492 — Allocation/VM&V Interface

Operation: This module interfaces with VM&V Control (IEFAB493) to process volume mount and verify requests (represented by VM&V request blocks build during generic and recovery allocation).

Entry from: IEFAB490.

Exit to: Caller.

Called routines: IEFAB4A0, IEFAB4FA, IEFAB4F0, IEFAB49A, IEFAB493, IEFAB498.

IEFAB493 — Volume Mount and Verify (VM&V) Control

Operation: This module controls unload, mount, and verify processing; it scans VM&V request blocks and calls unload control (IEFAB494), mount control (IEFAB495), and verify control (IEFAB496) to process the requests.

Entry from: IEFAB4A4, IEFAB429, IEFAB435, IEFAB441, IEFAB473, IEFAB477, IEFAB49B, IEFAB49C, IEFAB491, IEFAB492, or IEFABB416.

Exit to: Caller.

Called routines: IEFAB494, IEFAB495, IEFAB496.

IEFAB494 — Volume Unload Control

Operation: This module processes all requests to:

- Rewind or rewind and unload tape volumes
- Unload direct access volumes

Entry from: IEFAB493.

Exit to: Caller.

Called routines: IEFAB4E0, IEFAB4F3, IEFAB499, IEFENFFX.

IEFAB495 — Mount Control Routine

Operation: This module builds mount control blocks. For mass storage system (MSS) volume requests, this module interfaces with the 3850 mass storage system to mount the volumes. For non-MSS volume requests, this module issues mount WTO or WTOR messages.

Entry from: IEFAB493.

Exit to: Caller.

Called routines: IEFAB4E0, IEFAB498, IEFAB499.

IEFAB496 — Verify Control Routine

Operation: This module handles all verify device end and verify label requests by waiting for the specified units to become ready. In the case of verify label, this module also ensures that the requested volume has been mounted.

Entry from: IEFAB493.

Exit to: Caller.

Called routines: IEFAB49A, IEFAB49B, IEFAB498, IEFAB499.

IEFAB498 — MVCA Chain Processor

Operation: This module is invoked to perform one of the following functions:

- Search the MVCA chain for the MVCA for this allocation.
- Add an MVCA to the MVCA chain.
- Delete an MVCA from the MVCA chain.

Entry from: IEFAB4E2, IEFAB4E5, IEFAB4E8, IEFAB49A, IEFAB492, IEFAB495, IEFAB496.

Exit to: Caller.

Called routines: None.

IEFAB499 — VM&V WTO/WTOR Format Routine

Operation: This module builds WTO/R or multiline WTO/R parameter lists for MOUNT and UNLOAD and VERIFY messages using information supplied by the caller.

Entry from: IEFAB494, IEFAB495, IEFAB496.

Exit to: Caller.

Called routines: None.

IEFAB820 — TCTIOT Construction Interface

Operation: This module determines whether SMF accounting functions are required, initializes the TCT storage map, and calls IEFIB660 to build the TCTIOT.

Entry from: IEFSD263 via CALL (to IEFSMFAT, alias for IEFAB820).

Exit to: IEFSD263 via branch.

IEFAUINT — Assign/Unassign Initialization Service

Operation: This module assigns one or all on-line assignable devices. It is invoked by JES2 during initialization to assign all assignable devices with the ASGNALL function. It is also invoked by JES3 during main device scheduling initialization using the ASGNALL function for devices that are not JES3-managed and the ASGNONE function for the individual assigning of JES3-managed devices.

Entry from: JES2 or JES3.

Exit to: Caller.

Called routines: IEFAUSRV, IOSVSUCB, IEFAB4GB.

IEFAUSRV — Hardware Assign/Unassign Service

Operation: This module is invoked by IEFAUINT and other system components to assign a device, unassign a device, restore status of a device and build the CCWs to restore the assign status of a device.

Entry from: JES3, IEFAB488, IEFAB429, IECVIOPM, IEFAUINT, IECVGENA, IECVRRSV, IECVRSTI.

Exit to: Caller.

Called routines: None.

IEFBB4M1 — Job Status Messages

Operation: This module contains the message text for the allocation job status operator messages.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFBB4M2 — Allocation Step-Related Messages

Operation: This module contains the message text of allocation step-related error messages that are issued to the system output data set.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFBB4M3 — Allocation DD-Related Messages

Operation: This module contains the message text of the allocation DD-related error messages that are issued to the system output data set.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFBB4M4 — WTO Messages for IEFBB410

Operation: This module contains the message text of the unallocation job status messages.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFBB4M5 — Unallocation Messages

Operation: This module contains the message text of the unallocation error messages.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFBB401 — Initiator/Allocation Interface

Operation: This module controls the interface between the initiator and step allocation control. (It determines the path that a step or job will take through the initiator.)

Entry from: IEFSD162.

Exit to: Caller.

Called routines: IEEAB401, IEFAB4A6, IEFAB4FC, IEFAB4FD, IEFAB4FE, IEFAB4F3, IEFAB4F7, IEFBB402, IEFBB404.

IEFBB402 — Test EXEC Statement Condition Codes

Operation: This module determines, from the specifications of the COND parameter on the EXEC statement, whether the step should be bypassed or executed.

Entry from: IEFBB401.

Exit to: Caller.

Called routines: IEFAB4FD, IEFAB4F7.

IEFBB404 — Step Allocation Control

Operation: This module receives control when a job step is to be allocated; it controls the following functions:

- JFCB housekeeping
- Common allocation processing
- Reordering and compressing the TIOT
- Writing DD-related error messages

Entry from: IEFBB401.

Exit to: Caller.

Called routines: IEFAB4FC, IEFAB4FD, IEFAB4F3, IEFAB421, IEFAB451.

IEFBB410 — Initiator/Unallocation Interface

Operation: This module provides an interface between the initiator and step unallocation. SIOTs, JFCBs, and JFCBXs are located and chained, end-of-step messages are issued, and step unallocation is invoked.

Entry from: IEFSD164.

Exit to: Caller.

Called routines: IEEAB401, IEFAB4FC, IEFAB4FD, IEFAB4FE, IEFAB4F7, IEFAB412, IEFBB414, IEFBB416, IEFRPREP, IEFTB721.

IEFBB412 — Process Job Condition Codes

Operation: This module compares the return code from the step just executed with the dependency codes specified in the COND parameter on the JOB statement.

Entry from: IEFBB410.

Exit to: Caller.

Called routines: None.

IEFBB414 — Step Unallocation Control

Operation: This module creates the interface to Common Unallocation for the job step being terminated. This involves building a request block for each SIOT on the input SIOT chain and determining the processing required for each of these requests. Allocation messages are also built and issued if requested.

Entry from: IEFAB4DD, IEFBB410.

Exit to: Caller.

Called routines: IEFAB4A0, IEFAB4EE.

IEFBB416 — Job Unallocation

Operation: This module performs disposition processing for passed unreceived data sets and unloads volumes no longer needed by this job.

Entry from: IEFAB4DD, IEFBB410.

Exit to: Caller.

Called routines: IEFAB4A0, IEFAB4EB, IEFAB4F3, IEFAB4F7, IEFAB493.

IEFDB4A0 — Dynamic Unallocation Control Routine

Operation: IEFDB4A0 handles Dynamic Allocation (SVC99) requests for unallocation. It checks the validity of the request, determines what DSABs should be processed and what processing should be done for them (Unallocation or remove in-use), and invokes the necessary routine(s) to perform the processing.

Entry from: IEFDB400, IEFDB410.

Exit to: Caller.

Called routines: IEFDB4A1, IEFDB4FA, IEFDB4FC, IEFDB4FF, IEFDB460, IEFDB481.

IEFDB4A1 — Dynamic Unallocation Processor

Operation: This module receives control from either Unallocation Control (IEFDB4A0), or Remove In-use Processor (IEFDB481) to perform Unallocation Processing for DSABs pointed to by the input list of DSABPTRs.

Entry from: IEFDB4A0, IEFDB481.

Exit to: Caller.

Called routines: IEFAB4A0, IEFAB4DC, IEFAB4F7, IEFDB4F9.

IEFDB4D0 — DAIR - Dynamic Allocation Interface Routine

Operation: In a TSO environment, this module serves as an interface from various command processors to dynamic allocation.

Entry from: Various command processors and user programs.

Exit to: Caller.

Called routines: IEFDB400 (via SVC 99), IKJEHCIR (via LINK).

IEFDB4FA — Dsname Search Routine

Operation: This module searches the DSABs and JFCBs for the data set name requested by the user. If the dsname is allocated to the current task, the address of its DSAB is returned to the caller.

Entry from: IEFDB4A0, IEFDB470.

Exit to: Caller.

Called routines: None.

IEFDB4FB — Change DDNAME/JES3 Exit

Operation: This module interfaces with the JES3 subsystem whenever a DDNAME or relative position number change has been made by Dynamic Allocation.

Entry from: IEFDB410, IEFDB411, IEFDB450.

Exit to: Caller.

Called routines: None.

IEFDB4FC — Ddname Search Routine

Operation: This module searches the TIOT for a specific ddname. It begins its search at the starting DSAB address passed as input by the caller.

Entry from: IEFDB4A0, IEFDB4FD, IEFDB414, IEFDB450, IEFDB460, IEFDB470, IEFDB490.

Exit to: Caller.

Called routines: None.

IEFDB4FD — Ddname Generation Routine

Operation: This routine generates an eight character ddname.

Entry from: IEFDB410, IEFDB414.

Exit to: Caller.

Called routines: IEFDB4FC.

IEFDB4FE — Obtain DSORG Routine

Operation: This module determines the data set organization for the data set associated with the input DSAB.

Entry from: IEFDB410, IEFDB413, IEFDB470.

Exit to: Caller.

Called routines: None.

IEFDB4FF — Syntax Checker

Operation: This module determines whether the user has entered an invalid key, mutually exclusive keys, or duplicate keys. It also checks whether an invalid number of parameters has been entered with each key, and whether the length of each parameter value is valid.

Entry from: IEFDB4A0, IEFDB4F0, IEFDB450, IEFDB460, IEFDB470, IEFDD480, IEFDB490.

Exit to: Caller.

Called routines: None.

IEFDB4F0 — Key Verification

Operation: This module invokes the syntax checker (IEFDB4FF) to syntax check the text units with non-JDT-defined keys and invokes SJF (IEFSJCNL) to syntax check the text units with JDT-defined keys.

Entry from: IEFDB412.

Exit to: Caller.

Called Routines: IEFDB4FF and IEFSJCNL.

IEFDB4F8 — Allocate TCTIOT Control

Operation: Following a dynamic allocation, this module updates an existing TCTIOT or obtains and updates a new TCTIOT if the existing TCTIOT cannot be expanded.

Entry from: IEFDB4F9.

Exit to: Caller.

Called routines: None.

IEFDB4F9 — SMF Dynamic DD Routine

Operation: This module updates the TCTIOT to reflect any changes made to the TIOT as a result of a dynamic allocation, unallocation, concatenation, or deconcatenation. If the function is unallocation, concatenation, or deconcatenation, an SMF type 40 record is built and written.

Entry from: IEFDB4A1, IEFDB413, IEFDB450, IEFDB460, IEFDB400.

Exit to: Caller.

Called routines: IEEMB848, IEFDB4F8.

IEFDB400 — Dynamic Allocation Control Routine

Operation: To process the input request, this module routes control to one of the following functions: dsname allocation, ddname allocation, concatenation, deconcatenation, remove in-use attribute, information retrieval or unallocation.

Entry from: SVC interrupt handler.

Exit to: Caller.

Called routines: IEFAB4FE, IEFAB4F7, IEFDB4A0, IEFDB401, IEFDB410, IEFDB450, IEFDB460, IEFDB470, IEFDB480, IEFDB490, IEFDB4F9.

IEFDB401 — Dynamic Allocation Installation Exit

Operation: The Dynamic Allocation facility of the control program exits to this 'Validation Routine' before doing any processing on behalf of a Dynamic Allocation request. It is entered for all requests, foreground and background. This routine may test and modify the Dynamic Allocation input, and indicate through a return code whether processing is to continue or if the request is to be terminated. This IBM version arbitrarily accepts all requests.

Entry from: IEFDB400.

Exit to: Caller.

Called routines: None.

IEFDB402 — Dynamic Allocation ESTAE Exit

Operation: This module performs clean-up processing when the task under which SVC 99 is executing abnormally terminates.

Entry from: IEFAB4ED.

Exit to: Caller.

Called routines: IEFDB418.

IEFDB403 — TCTIOT FRR Routine

Operation: This exit routine receives control if an ABEND occurred while TCTIOT updating was taking place. This module sets to zero the pointer to the TCTIOT if it is unuseable and frees TCTIOT storage.

Entry from: RTM (recovery termination management).

Exit to: Caller.

Called routines: None.

IEFDB410 — Allocate Function Control

Operation: This module is the control routine for the dynamic allocation of a data set.

Entry from: IEFDB400.

Exit to: Caller.

Called routines: IEFAB4F7, IEFDB4A0, IEFDB4FD, IEFDB4FE, IEFDB411, IEFDB412, IEFDB413, IEFDB460, IEFDB4FB.

IEFDB411 — Dynamic Allocation Convert Routine

Operation: This module modifies an existing allocation to satisfy the user's request.

Entry from: IEFDB410.

Exit to: Caller.

Called routines: IEFAB4DC, IEFAB4E9, IEFDB417.

IEFDB412 — Dynamic Allocation Function Validity Checker

Operation: This module validity checks the allocation function's input text against invalid, duplicate, or mutually exclusive keys; invalid number of keyword parameters or parameter lengths; absence of mutually inclusive parameters; invalid parameter values; and improper authorization to request a subsystem data set.

Entry from: IEFDB410.

Exit to: Caller.

Called routines: IEFDB4F0.

IEFDB413 — Normal Dynamic Allocation Control

Operation: This routine controls the processing for a normal dynamic allocation request.

Entry from: IEFDB410.

Exit to: Caller.

Called routines: IEFAB4A0, IEFAB4DC, IEFAB4FC, IEFAB4F7, IEFAB421, IEFAB451, IEFDB4FE, IEFDB4F9, IEFDB414, IEFDB418.

IEFDB414 — Build SWA Tables For Dynamic Allocation Request

Operation: This module creates and sets up a SIOT, JFCB, and if necessary, JFCBX(s), and/or a SSWA.

Entry from: IEFDB413.

Exit to: Caller.

Called routines: IEFAB4F7, IEFDB4FC, IEFDB4FD, IEFDB417, IEFSJCNL.

IEFDB417 — Dynamic Allocation JFCB - DCB Field Update

Operation: This module updates fields in the JFCB for DCB keys specified in the user's request.

Entry from: IEFDB411, IEFDB414.

Exit to: Caller.

Called routines: None.

IEFDB418 — Dynamic Allocation Normal Error Subroutine

Operation: This module serves as a subroutine for clean up processing if an error occurs in dynamic allocation. It releases resources acquired during processing of a dynamic allocation request and ensures that the 'WRITE-EPA' chain is written to SWA.

Entry from: IEFDB402, IEFDB413.

Exit to: Caller.

Called routines: IEFAB4A0, IEFAB4DC, IEFAB4F7.

IEFDB450 — Concatenation Routine

Operation: This module concatenates existing allocations in the order in which they are specified in the input text units.

Entry from: IEFDB400.

Exit to: Caller.

Called routines: IEFAB4FC, IEFAB4F7, IEFDB4FB, IEFDB4FC, IEFDB4FF, IEFDB4F9.

IEFDB460 — Deconcatenation Routine

Operation: This module disassociates members of a dynamically concatenated group by restoring the pre-concatenation ddnames to the members and removing the 'Dynamically Concatenated' (DSABDCAT) and 'Member of a Concatenated Group' (DSABCATM) attributes from the respective DSABs.

Entry from: IEFDB4A0, IEFDB400, IEFDB410.

Exit to: Caller.

Called routines: IEFAB4FC, IEFDB4FF, IEFDB4F9.

IEFDB470 — Dynamic Information Retrieval

Operation: This module provides the user with information about his current allocation environment.

Entry from: IEFDB400.

Exit to: Caller.

Called routines: IEFDB4FA, IEFDB4FC, IEFDB4FE, IEFDB4FF.

IEFDB480 — Remove In-Use Control Routine

Operation: This module handles dynamic allocation (SVC 99) requests for remove in-use processing. It validity checks the request, determines what DSABs should be processed, and invokes the remove in-use processor (IEFDB481).

Entry from: IEFDB400.

Exit to: Caller.

Called routines: IEFDB4FF, IEFDB481.

IEFDB481 — Remove In-Use Processor

Operation: This module performs remove in-use processing for the DSABs pointed to by the input list.

Entry from: IEFDB4A0, IEFDB480.

Exit to: Caller.

Called routines: IEFDB4A1.

IEFDB490 — Ddname Allocation

Operation: This module assigns the in-use attribute to the resource associated with the specified ddname.

Entry from: IEFDB400.

Exit to: Caller.

Called routines: IEFDB4FC, IEFDB4FF.

IEFDPOST — Device Interrupt Routine

Operation: This module is invoked by IOS to post tasks waiting for devices to become ready.

Entry from: IOS (input/output supervisor).

Exit to: Caller.

Called routines: IEAOPT01.

IEFDSLST — Data Set Enqueue Parameter List Builder

Operation: This module builds a data set enqueue parameter list for each job; to do this, it uses input passed by IEFDSTBL.

Entry from: IEFSD161 via branch.

Exit to: IEFSD161 via branch.

IEFDSTBL — Data Set Tree Builder

Operation: This module eliminates duplicate data set names from the data set enqueue parameter list that exists for each job.

Entry from: IEFSD161 via branch.

Exit to: IEFSD161 via branch.

IEFEB400 — EDT Verification Routine

Operation: This module verifies that the devices defined by the EDT contained in the input data set match the devices currently existing in the nucleus.

Entry from: Program manager.

Exit to: Caller.

IEFENFDM — Event Notification Control Table

Operation: This module contains the event notification control table (IEFENFCT).

Entry from: Not applicable (non-executable module).

Exit to: None.

IEFENFFX — Event Notification Request Router

Operation: This module checks the event parameter list and determines if the request is synchronous or asynchronous.

Entry from: Event notification user.

Exit to: Caller.

Called routines: IEFENFNM.

IEFENFNM — Event Notification Mainline

Operation: Thisbmodule processes all event notification listen, signal, and delete listen requests.

Entry from: IEFENFWT, IEFENFFX.

Exit to: Caller.

IEFENFWT — Event Notification Wait Routine

Operation: This module handles processing of asynchronous event notification requests.

Entry from: IEEMB860 via ATTACH and IEFENFFX via POST.

Exit to: None, this is a never ending task.

Called routines: IEFENFNM.

IEFHB4I1 — Allocation Address Space Initialization

See *OS/VS2 System Initialization Logic* for a module description of IEFHB4I1.

IEFHB4I2 — Display Allocation Scavenge Routine

See *OS/VS2 System Initialization Logic* for a module description of IEFHB4I2.

IEFHB410 — Display Allocation Tables Manager

Operation: This module handles all updates to the display allocation tables in the allocation address space (ALLOCAS). IEFHB410 executes in cross memory mode. It is entered by means of a program call (PC) to perform one or more of the following functions.

- Updates the DALT use count field
- Returns the DALT use count for a unit
- Returns the DALT use count for one address space and one unit
- Clears the DALT for an address space

Entry from: IEFAB4A4, IEFAB4E5, IEFAB4E6, IEFAB434, IEFAB435, IEFHB4I2, IEFAB477.

Exit to: Caller by means of a program transfer (PT).

IEFIB600 — Initiator/Subsystem SWA Interface

Operation: This module builds the JSCB chain for a job, invokes the interpreter, and builds a CSCB for a job that is not a started task or LOGON.

Entry from: The master subsystem or the job entry subsystem via LINK.

Exit to: Caller via branch.

IEFIB605 — SWA Reconstruction Module

Operation: This module reconstructs SWA for restarted jobs by invoking the SWA merge routine and the data set description record processor.

Entry from: IEFIB605 via CALL.

Exit to: Caller via branch.

IEFIB620 — ESTAE Exit Routine

Operation: Depending upon what type of error caused this module to receive control, it records the error and then either takes a dump or prepares for a retry.

Entry from: IEAVTAS1 via a SYNCH macro.

Exit to: IEAVTAS1 via branch.

IEFIB621 — Initiator Task Recovery Retry Routine

Operation: This module attempts to resume normal initiator processing after a program check, an ABEND, or machine check has occurred, or after an operator has pushed the RESTART key.

Entry from: This module receives control and executes as a result of an RB (request block) created by the recovery termination management routines.

Exit to: IEFSD164 via branch.

IEFIB645 — SWA Create Exit Routine

Operation: This module receives control whenever an error has occurred during interpreter, restart, or SWA create interface processing. It records the error on the system error record data set and then attempts a retry.

Entry from: Recovery termination management via LINK.

Exit to: Caller via branch.

IEFIB650 — Initiator Message Module

IEFIB660 — Build TCTIOT Routine

Operation: This module constructs a TCTIOT based on the current TIOT. If a TCTIOT exists on entry to this module, it is freed. This situation could occur if dynamic allocation obtained a TCTIOT prior to initiation.

Entry from: IEFAB820.

Exit to: Caller.

Called routines: None.

IEFICATL — Initiator Interface to Allocate Catalog Control

Operation: This module determines the data set names of each catalog to be used by a jobstep/task and then invokes IEFAB4F5 to open the catalogs.

Entry from: IEFSD162 via BALR.

Exit to: IEFSD162 via branch.

IEFICPUA — Assignment of CPU Task Affinity

Operation: This module indicates to the calling module which processor(s) can be used to run a jobstep/task; it also notifies the calling routine if the required processor(s) is not on line.

Entry from: IEFSD101 for the first jobstep in a job and from IEFSD101 for subsequent steps.

Exit to: The calling module via branch.

IEFIIC — Initiator Interface Control

Operation: This module puts the initiator routines into supervisor state and builds the initiator's SSIB, SSOB header, entrance, options, and exit list (IEL).

Entry from: IEFSD263 via ATTACH.

Exit to: IEFSD060 via branch.

IEFIMASK — Conversion of Bit Mask

Operation: This routine converts bit positions in a string to hexadecimal characters for inclusion in some initiator messages.

Entry from: IEFSD101 or IEFSD161 via branch.

Exit to: The calling module via branch.

IEFIRECM — Initiator Resource Manager

Operation: This module cleans up resources used by the initiator in a terminating address space. This routine deletes all CSCBs associated with this address space from the CSCB chain.

Entry from: RTM (via address in CVTIRECM field).

Exit to: RTM.

IEFISEXR — Initiator Subsystem ESTAE Exit Routine

Operation: This module receives control after an abnormal situation occurs in the initiator or subsystem interface resource managers. It issues a DEQ for the CSCB chain and makes an error entry in the LOGREC data set.

Entry from: RTM (via address in CVTJRECM field).

Exit to: RTM.

IEFI922B — Builder of 922 Completion Code Interface

Operation: This routine builds an interface to module IEFSD164 to enable the termination of a jobstep/task which has already successfully completed allocation processing.

Entry from: IEFIB621 via CALL.

Exit to: IEFIB621 via branch.

IEFJACTL — Pseudo Access Method Control

Operation: This routine, a part of the master subsystem, provides a data manipulation service for the converter and interpreter at a time when no access method services are available via the RPL/ACB interface.

Entry from: The converter or interpreter.

Exit to: Caller.

Error exit: To caller's ESTAE routine with a user abend code X'0B1' or X'0B3'.

IEFJCDLT — Storage Deletion Routine

Operation: This routine, a part of the master subsystem, frees the storage no longer needed by the master subsystem after creating SWA control blocks for a task.

Entry from: IEFJJOBS.

Exit to: Caller.

IEFJCNTL — JCLS to SWA Conversion

Operation: This routine, a part of the master subsystem, converts a JCLS chain to SWA control blocks by invoking the converter, then the interpreter (via the SWA-create interface).

Entry from: IEFJJOBS.

Exit to: Caller.

Error exit: To caller's ESTAE routine with a user abend code X'0B1', X'0B4', or X'0B5'.

IEFJDIRD — Pseudo Access Method Direct Read

Operation: This routine, a part of the master subsystem, moves the specified record into a specified buffer so that the record can be updated.

Entry from: The converter via the RPL/ACB interface.

Exit to: Caller.

IEFJDSNA — Data Set Name Assignment

Operation: This routine, a part of the master subsystem, assigns a data set name to each SYSOUT data set specified in the master JCL or in the JCL used to start a job entry subsystem.

Entry from: Allocation.

Exit to: Caller.

IEFJDWRT — Pseudo Access Method Direct Write

Operation: This routine, a part of the master subsystem, overlays an old record with an updated record from a buffer.

Entry from: The converter via the RPL/ACB interface.

Exit to: Caller.

IEFJJCLS — JCL to JCLS Conversion

Operation: This routine, a part of the master subsystem, converts the master scheduler JCL (MSTRJCL) to a JCLS chain.

Entry from: IEFJJOBS.

Exit to: Caller.

Error exit: To caller's ESTAE routine with a user abend code X'0B1'.

IEFJJOBS — Subsystem Initiation

Operation: This routine, a part of the master subsystem, controls the creation of SWA control blocks when the master scheduler or a job entry subsystem is being started.

Entry from: The initiator via the subsystem interface.

Exit to: The initiator.

Error exit: To caller's ESTAE routine with a user abend code X'0B1'.

IEFJJTRM — Subsystem Job Termination

Operation: This routine, a part of the master subsystem, is a dummy routine. It replaces the normal job termination function when the job being terminated is a subsystem.

Entry from: A system routine via the IEFSSREQ macro.

Exit to: The system routine.

IEFJRASP — Common Request Router

Operation: This routine, a part of the master subsystem, handles requests that require processing for all active subsystems except the master subsystem.

Entry from: A system routine via the subsystem interface.

Exit to: The system routine.

IEFJREAD — Pseudo Access Method Sequential Read

Operation: This routine, a part of the master subsystem, moves a record from a chain of records to a buffer and prepares for the next record to be read from the chain.

Entry from: The converter or interpreter via the RPL/ACB interface.

Exit to: Caller.

IEFJRECF — Subsystem Interface Resource Manager

IEFJRECF is a secondary entry point of IEFJRECM. See the IEFJRECM module description.

IEFJRECM — Subsystem Interface Resource Manager

Operation: This module checks the resource manager parameter list for task or address space termination indicators. It then builds either a task or address space SSOB extension. If an address space is terminating, it also builds a list of jobnames associated with the terminating address space. It then issues the IEFSSREQ macro to communicate end-of-task or end-of-address space to all active subsystems.

RTM enters this module at secondary entry point IEFJRECF to notify all active subsystems of an end-of-task. At this entry point, the module builds the end-of-task SSOB extension and issues IEFSSREQ macro. IEFJRECF is located before IEFJRECM in RTM's list of end-of-task resource managers.

Entry from:

For entry point IEFJRECM: RTM using the address in the JESRECM field of the JESCT.

For entry point IEFJRECF: RTM using the address in the JESRECF field of the JESCT.

Exit to: RTM.

IEFJSBLD — Subsystem Service Routine

See *OS/VS2 System Initialization Logic* for a module description of IEFJSBLD.

IEFJSIMM — Subsystem Initialization and Parmlib Messages

Operation: This module contains messages for subsystem initialization and parmlib processing. These messages are accessed and issued by IEFJSIMW.

Entry from: None.

Exit to: None.

IEFJSDTN — Subsystem Determination

Operation: This routine, a part of the master subsystem, determines if the task being started is a subsystem.

Entry from: The initiator via the subsystem interface.

Exit to: Return to the initiator.

IEFJSIMW — Subsystem Initialization and Parmlib Processing Message Writer

Operation: This module issues messages, which are contained in IEFJSIMM. It issues these messages during subsystem initialization (for IEFJSBLD and IEFJSIN2) and during the processing of the SET MPF command (for IEECB805).

Entry from: IEECB805 and modules used during subsystem initialization (IEFJSBLD and IEFJSIN2). (See *OS/VS2 System Initialization Logic.*)

Exit to: Caller.

IEFJSINT — Subsystem Interface Initialization

See *OS/VS2 System Initialization Logic* for a module description of IEFJSINT.

IEFJSIN2 — Subsystem Initialization

See *OS/VS2 System Initialization Logic* for a module description of IEFJSIN2.

IEFJSREQ — Subsystem Interface

Operation: IEFJSREQ handles requests for subsystem interface services from the IEFSSREQ macro. It determines which routine of which subsystem is to receive control and then passes control to that routine via a branch instruction.

Entry from: A system routine via the IEFSSREQ macro.

Exit to: The subsystem routine to perform the requested function. (See the figure "Subsystem Interface Summary," for the module names.)

Error exit: Caller with nonzero return code.

IEFJSVEC — Subsystem Vector Table Service

Operation: This module provides an interface for calling the subsystem service routine (IEFJSBLD) which creates a subsystem vector table (SSVT) or modifies an existing one.

Entry from: Any subsystem.

Exit to: Caller.

IEFJSWT — STC Write JCL Routine

Operation: This module writes the internal JCL text for a START, MOUNT, or LOGON command.

Entry from: IEESB605 via BALR.

Exit to: Caller via branch.

IEFJWRTE — Pseudo Access Method Sequential Write

Operation: This routine, a part of the master subsystem, adds a record to a chain of records by obtaining storage, moving the new record to that storage area, and chaining the previous record to the new one.

Entry from: The converter via the RPL/ACB interface.

Exit to: The converter.

IEFJWTOM — Subsystem Initiation Message Writer

Operation: This routine, a part of the master subsystem, writes a hard copy of the converter, interpreter, and allocation error messages (including the JCL card images in error) for tasks being started via the master subsystem.

Entry from: The converter, interpreter, or allocation via the RPL/ACB interface.

Exit to: Caller.

IEFNB901 — Data Management Interpreter Exit Routine

Operation: This module builds a parameter list for the data management interpreter routine and then passes control to that module which processes the AMP keyword parameter.

Entry from: From IEFVDA via branch.

Exit to: Data management interpreter routine via LOAD and BALR, caller via branch.

IEFNB903 — Interpreter Initialization Routine

Operation: This routine obtains storage for and partially initializes the interpreter work area (IWA), its I/O buffer, the JMR, and the DSENQ table. It also establishes an ESTAE environment for the interpreter.

Entry from: IEFIB600 via LINK.

Exit to: IEFVHE via branch or to the journal write routine via branch for restarted jobs.

IEFQB550 — SWA Manager Move Mode Functions

Operation: This module assigns, writes, reads, or deletes scheduler work area (SWA) blocks in move mode.

Entry from: Initiator/terminator, scheduler restart, converter/interpreter, IEFQB580 or IEFQB585.

Exit to: Caller via branch.

IEFQB555 — SWA Manager Locate Mode Functions

Operation: This module assigns, writes, reads, or deletes scheduler work area (SWA) blocks in locate mode.

Entry from: Initiator/terminator or scheduler restart or allocation or the scheduler JCL facility via SWAREQ macro instruction.

Exit to: Caller via branch.

IEFQB580 — QMNGRIO Macro Interface Handler

Operation: This module provides an interface to SWA manager functions from data management routines that use the QMNGRIO macro instruction. IEFQB580 builds and initializes a queue management parameter area (QMPA) for the module requesting SWA manager functions.

Entry from: Data management routines via QMNGRIO macro instruction.

Exit to: IEFQB550 via BALR.

IEFQB585 — SWA Manager Interface Module

Operation: This module intercepts any references to previously existing queue manager modules. IEFQB585 inserts the appropriate function code in the queue management parameter area (QMPA) on behalf of the requesting module.

Entry from: Modules that reference previously existing queue manager modules.

Exit to: IEFQB550 via BALR.

IEFQMWR — Allocation/Termination Communication Area

Operation: This non-executable CSECT contains the address of the nucleus-resident update DDR count routine (IEFAB438) and a pointer to the first MVCA on the MVCA chain. This area is called the allocation/termination communication area (ATCA) and is mapped by macro IEFZB432.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFRPREP — Restart Preparation Routine

Operation: This module determines if an ABENDed jobstep task can be restarted.

Entry from: IEFBB410.

Exit to: IEFBB410.

IEFSDPPT — Program Properties Table

IEFSD060 (IEFSD160) — Initiator Control Initialization

Operation: This module builds the linkage control block (LCT) for a jobstep/task to be processed by the initiator and a STAE parameter list (STEPL) for the initiator's STAE exit routine.

Entry from: IEESB605 via LINK for LOGON, START, and MOUNT commands; IEFIIC via branch for normal initiator processing; IEEVIPL via ATTACH for master scheduler initialization.

Exit to: IEFSD061 via branch.

IEFSD061 (IEFSD161) — Job Select Routine

Operation: This routine requests jobs for the initiator from the job entry subsystem and begins STOP processing as required for started tasks.

For a selected job to be warmstarted, IEFSD061 sets up for warmstart processing; for all other selected jobs this module assigns special protect keys and processes data set integrity, if required.

Entry from: IEFSD160 via branch for job selection or from IEFSD166 for job selection or stop processing.

Exit to: A generalized initiator exit via branch, LINK, or XCTL for internal stop processing; to IEFSD164 via branch for warmstart; to IEFSD101 for normal initiator processing.

IEFSD062 (IEFSD162) — Device Allocation Interface Routine

Operation: This module invokes device allocation for a jobstep/started task, opens a JOBLIB, STEPLIB, or FETCHLIB required by the jobstep/started task, and completes assignments of special properties.

Entry from: IEFSD102 via branch.

Exit to: IEFSD103 via BALR, to IEFSD164 via BALR if an error occurred during allocation or OPEN catalog processing.

IEFSD064 (IEFSD164) — Step Delete Routine

Operation: When a jobstep/task completes processing, this module closes the JOBLIB, STEPLIB, or FETCHLIB DCBs, performs job and jobstep timing calculations, and builds a dummy TCB to be used by termination routines.

Entry from: IEFSD161 via branch for warmstarted jobs; IEFSD162 via branch when an error occurs in allocation for a jobstep/started task; IEFSD263 via branch when a jobstep/started task completes processing; IEFIB621 to retry after ESTAE processing.

Exit to: IEFSD101 via branch to initiate the next step in a job; IEFSD166 via branch to delete a job when all jobsteps have completed or to suspend a job, if necessary.

IEFSD066 (IEFSD166) — Job Delete Routine

Operation: This module deletes a job and its associated control blocks when the job completes processing or re-enqueues a job when necessary.

Entry from: IEFSD164 via branch.

Exit to: IEFSD161 via branch.

IEFSD101 — PPT Scan

Operation: This routine scans the program properties table (PPT) and assigns special properties to a jobstep/started task as indicated; it also builds a GETPART work table if one is required by the jobstep/started task.

Entry from: IEFSD161 via branch for the first jobstep in job and from IEFSD164 via branch for subsequent jobsteps.

Exit to: IEFSD102 via branch.

IEFSD102 — Data Set Enqueue

Operation: This routine enqueues on the data sets required for an entire job.

Entry from: IEFSD101 via branch.

Exit to: IEFSD162 via branch.

IEFSD103 — ATTACH Interface Routine

Operation: This module builds the ATTACH parameter list.

Entry from: IEFSD162 via branch.

Exit to: IEFSD263 via branch.

IEFSD160 — See IEFSD060

IEFSD161 — See IEFSD061

IEFSD162 — See IEFSD062

IEFSD164 — See IEFSD064

IEFSD263 — Initiator ATTACH Module

Operation: This module gets a region for a jobstep/started task, if one is required, attaches the jobstep/started task, and then waits for an end-of-task or cancel ECB to be posted. When the appropriate ECBs are posted, this module detaches the jobstep/started task.

Entry from: IEFSD103 via branch.

Exit to: IEFSD164 via branch.

IEFSJBLD — Scheduler JCL Facility Build SWB Routine

Operation: This module builds a scheduler work block (SWB).

Callers: IEFSJWRT.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point BLDRETRY: This routine performs clean up processing when an ABEND occurs during SJF build processing.

Callers: Recovery termination manager (RTM).

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: SWA manager locate mode (IEFQB555) and an alternate SWA manager routine.

IEFSJCNL — Scheduler JCL Facility (SJF) Control Routine

Operation: This routine is the entry point for all scheduler JCL facility (SJF) requests. It performs common initial processing for the SJF function, routes the request to the specified SJF function, and upon return, performs common clean up processing.

Callers: Any routine issuing the SJF request macro (SJFREQ).

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point RECOVERY: This routine is used to recover from an error that caused the exit to RTM.

Callers: RTM.

Exit-Normal: Returns to the mainline clean up processing if the ABEND occurred while SJF was processing a request.

Exit-Error: Percolates to the caller's recovery routine if the ABEND did not occur while SJF was processing or if a previous ABEND occurred.

Entry Point RECCLEAN: This routine performs clean up processing when an ABEND occurs during the SJF control routine's processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: IEFSJDEF, IEFSJDEL, IEFSJEXT, IEFSJFND, IEFSJGET, IEFSJINT, IEFSJJDV, IEFSJPUT, IEFSJRET, IEFSJUPD, and IEFSJWRT.

IEFSJDEF — Scheduler JCL Facility Define JDVT Routine

Operation: This routine creates a JCL definition vector table (JDVT) and adds this table to the JDVT chain anchored off the JES control table (JESCT).

Callers: IEFSJCNL and IEFSJINT.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point DEFRETRY: This routine performs clean up processing when an ABEND occurs during SJF define JDVT processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to RTM.

External References: No routines.

IEFSJDEL — Scheduler JCL Facility Delete SWB Chain Routine

Operation: This routine deletes a scheduler work block (SWB) chain.

Callers: IEFSJCNL and IEFSJUPD.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point DELRETRY: This routine performs clean up processing when an ABEND occurs during SJF delete processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: SWA manager and the journal write routine (IEFXB500).

IEFSJEXT — Scheduler JCL Facility (SJF) Extract Routine

Operation: This routine extracts information from the JCL definition table (JDT) associated with a verb, a verb and keyword, a verb and key, or subparameters of a keyword or key.

Callers: IEFSJCNL, IEFSJUPD, and IEFSJRET.

Exit-Normal: Return to caller.

Exit-Error: Return to caller.

Entry Point EXTRETRY: This routine performs clean up processing when an ABEND occurs during the SJF extract routine's processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: IEFSJJDV.

IEFSJFND — Scheduler JCL Facility Find SWB Chain Routine

Operation: This routine locates a scheduler work block (SWB) chain at a particular level of the scheduler work area (SWA) structure.

Callers: IEFSJCNL and IEFSJUPD.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point FNDRETRY: This routine performs clean up processing when an ABEND occurs during SJF find SWB processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: No routines.

IEFSJGET — Scheduler JCL Facility Get SWB Chain Routine.

Operation: This routine copies selected keywords from a SWB chain in text unit format into a storage area specified by the caller. The keywords obtained are those whose JDT flags match the qualifier flags set in the input parameter list.

Callers: IEFSJCNL.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point GETRETRY: This routine performs clean up processing when a ABEND occurs during SJF get processing.

Callers: RTM.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

External References: IEFSJJDV and IEFSJRET.

IEFSJINT — Scheduler JCL Facility JDVT Initialization Routine

Operation: This routine builds the system default JCL definition vector table.

Callers: IEFSJCNL.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point INTRETRY: This routine performs clean up processing when an ABEND occurs during SJF JDVT initialization processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: IEFSJDEF and IEFJSIMW.

IEFSJJDV — Scheduler JCL Facility (SJF) Find JDVT Routine

Operation: This routine locates a JCL definition vector table (JDVT).

Callers: IEFSJCNL, IEFSJEXT and IEFSJGET.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point JDVRETRY: This routine performs clean up processing when an ABEND occurs during SJF find JDVT processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: No routines.

IEFSJPUT — Scheduler JCL Facility Put SWB Routine

Operation: This routine rebuilds a SWB chain from SWB keyword data found in text unit format.

Callers: IEFSJCNL.

Exit-Normal: Return to caller.

Exit-Error: Return to caller.

Entry Point PUTRETRY: This routine performs clean up processing when an ABEND occurs during SJF put processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: IEFSJUPD and IEFSJWRT.

IEFSJRET — Scheduler JCL Facility (SJF) Retrieve Routine

Operation: This routine retrieves parameter information from a scheduler work block (SWB) chain associated with a keyword or keywords for a particular verb and label, and uses that information to build text units.

Callers: IEFSJCNL and IEFSJGET.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point RETRETRY: This routine performs clean up processing when an ABEND occurs during SJF retrieve processing.

Callers: RTM.

Exit-Normal: None

Exit-Error: Return to the caller.

External References: IEFSJEXT.

IEFSJUPD — Scheduler JCL Facility (SJF) Update Routine

Operation: This routine verifies the text units specified by the caller and if requested, updates the SWB chain with the information specified in text unit format.

Callers: IEFSJCNL and IEFSJPUT.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point UPDRETRY: This routine performs clean up processing when an ABEND occurs during the SJF update routine's processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: IEFSJDEL, IEFSJEXT, IEFSJFND, IEFSJWRT, and IEFXB500.

IEFSJWRT — Scheduler JCL Facility (SJF) Write SWB Routine

Operation: This routine locates a specific SWB and updates its data portion.

Callers: IEFSJCNL, IEFSJUPD, and IEFSJPUT.

Exit-Normal: Return to the caller.

Exit-Error: Return to the caller.

Entry Point WRTRETRY: This routine performs clean up processing when a ABEND occurs during SJF write processing.

Callers: RTM.

Exit-Normal: None.

Exit-Error: Return to the caller.

External References: IEFSJBLD.

IEFSMFIE — SMF Initialization Exit Support Module

Operation: This module constructs a timing control table (TCT) and supports user's job initiation exit routines and step initiation exit routines. It also issues SMF job commencement record type 20 and/or type 30.

Entry from: IEFSD101 via BALR.

Exit to: IEFSD101 via branch.

Called routines: IEFUSI, IEFUJI, IEFSMF30, and IEFSMF32.

IEFSMF30 — Type 30 Setup Record for SMF

Operation: This module builds the model for type 30 SMF records. IEFSMFIE calls the module at job start; IEFTB721 calls it at step termination if a warm start occurred and previous records were lost.

Entry from: IEFSMFIE and IEFTB721.

Exit to: Caller.

IEFSMF32 — Type 32 Record Setup for SMF

Operation: This module is called at job start by IEFSMFIE to build the model for type 32 SMF records.

Entry from: IEFSMFIE.

Exit to: Caller.

IEFTB721 — SMF Job and Step Termination Processor

Operation: This module provides the user with job and step information after the job or step completes. The module contains an ESTAE routine to maintain control block integrity should a user error occur in the user exit, IEFACTRT.

Entry from: IEFSD263.

Exit to: Caller.

Called routines: IEFTB722, IEFSMF30, IEFTB727, IEFTB726, IEFACTRT.

IEFTB722 — SMF Job and Step Termination Record Assembler

Operation: This module assembles job and step information into an SMF record format (type 4, 5, 34, or 35) after the job or step completes.

Entry from: IEFTB721.

Exit to: Caller.

IEFTB723 — SMF Exit Write-to-Programmer Processor

Operation: This module provides an interface that can be used by IEFACTRT accounting exits to issue messages to the programmer.

Entry from: May be called via BALR from IEFACTRT. (IEFYS is an alternate entry name for IEFTB723).

Exit to: IEFACTRT via branch.

IEFTB726 — Type 30 SMF Record Builder

Operation: This module builds a type 30 SMF record at step end.

Entry from: IEFTB721.

Exit to: Caller.

Caller routines: IEFTB728.

IEFTB727 — Type 32 SMF Record Builder

Operation: This module builds and writes the step type 32 record at step end or interval expiration.

Entry from: IEFTB721 and IEEMB836.

Exit to: Caller.

IEFTB728 — Delta Type 30 SMF Record Builder

Operation: This module builds a type 30 SMF record which contains data representing changes made since the last record or step start, whichever is later.

Entry from: IEEMB836 and IEFTB726.

Exit to: Caller.

IEFUJI — User Job Initiation Exit Routine

IEFUJV — Converter/Interpreter SMF User Exit Routine

IEFUSI — User Step Initiation Exit Routine

IEFUTL — User Time Limit Exit Routine

IEFVDA — Interpreter DD Card Processor

Operation: This routine creates and initializes the control blocks for a DD statement. It also checks the validity of DD keyword values and then sets a job fail indicator if it finds an invalid keyword value.

Entry from: IEFVHE via branch.

Exit to: IEFVHE via branch.

IEFVDBSD — Interpreter DSENQ Table Processor

Operation: This module creates a DSENQ table for all explicitly named (by DSNAME parameter) data sets and marks each as exclusive or shared.

Entry from: IEFVDA or IEFVEA via branch.

Exit to: Caller via branch.

IEFVEA — Interpreter EXEC Statement Processor

Operation: This routine creates a step control table (SCT) for an EXEC statement and the override and refer-back tables required for the step. It also chains the SIOTs and JFCBs for a job library, job catalog, and/or concatenation of data sets for the step.

Entry from: IEFVHE via branch.

Exit to: Caller via branch.

IEFVFA — Converter Scan Routine

Operation: This module scans the JCL card images for syntax errors and merges JCL from the JCL data set and a cataloged or in-stream procedure if required. It then converts each JCL statement into internal JCL text.

Entry from: IEFVHEB via branch.

Exit to: IEVHF via branch.

IEFVFB — Converter Symbolic Parameter Routine

Operation: This module assigns values specified on an EXEC procedure statement or PROC statement to symbolic parameters that appear on JCL statements in cataloged or in-stream procedures.

Entry from: IEFVFA via BALR.

Exit to: Caller via branch.

IEFVGK — Interpreter GET Key/Positional Utility Routine

Operation: This routine sets a pointer to the next JCL keyword or positional parameter in the internal text buffer for the JCL statement processor routines, IEFVJA, IEFVEA, IEFVDA, or IEFVJDTI.

Entry from: IEFVJA via branch, or from IEFVEA via branch, or from IEFVDA via branch, or from IEFVJDTV via branch.

Exit to: Caller via branch.

IEFVGM — Converter/Interpreter Message Module

Operation: This module puts error messages for the entire converter/interpreter into the message data set, and/or any required JCL statements to the list data set.

Entry from: IEFVJA via branch, or from IEFVEA via branch, or from IEFVDA via branch, or from IEFVJDTV via branch, or from IEFVJDTI via branch.

Exit to: Caller via branch.

IEFVGT — Converter Test and Store Utility Routine

Operation: This module performs miscellaneous functions for the JCL statement processors, IEFVJA, IEFVEA, and IEFVDA, based on indicators in the parameter descriptor table passed to it by the processor routines.

Entry from: IEFVJA, IEFVEA, or IEFVDA via branch.

Exit to: Caller via branch.

IEFVHA — Converter GET Routine

Operation: This module reads into an input buffer JCL statements from the JCL data set, from the procedure library, and/or from in-stream procedures.

Entry from: IEFVHC or IEFVIND via branch.

Exit to: IEFVHC via branch.

IEFVHC — Converter Comment or Continuation Validation Routine

Operation: This routine determines whether a valid comment or continuation is indicated on a JCL statement.

Entry from: IEFVHA via branch.

Exit to: to IEFVHEB via branch, if a continuation was expected and was or was not received; to IEFVHA via branch, if a comment was received; to IEFVHCB, via branch, if no continuation was expected.

IEFVHCB — Converter Verb Identifier Routine

Operation: This routine identifies the verb on a JCL statement and merges JCL statements from the JCL data set and the procedure library.

Entry from: IEFVHC, IEFVHL, or IEFVHF via branch.

Exit to: IEFVHA, via branch, when the JCL statement does not contain a verb; to IEFVHA, via branch, when the procedure library buffer must be primed; to IEFVHEB, via branch, when the JCL statement contains a JOB, EXEC, DD, or NULL verb; to IEFVHM, via branch, when the verb is unrecognizable; to IEFVINA, via branch, when a PROC verb is found.

IEFVHE — Interpreter GET and Route Routine

Operation: This routine gets JCL text strings from the internal text data set and routes them to the appropriate processor. It also determines when the SCT for a step or the JCT for a job should be written into SWA.

Entry from: IEFNB903 via branch.

Exit to: IEFVJA via branch when a JOB statement text string is encountered; to IEFVEA via branch when an EXEC statement text string is encountered; to IEFVDA via branch when a DD statement text string is encountered; to IEFVJDTV when a JDT-defined statement text string is encountered; to IEFVHH when an SCT or JCT is to be written into SWA.

IEFVHEB — Converter Pre-scan Routine

Operation: This routine performs checkpoint/restart functions, enters JCL statements into the list data set, and initializes the JMR for SMF processing.

Entry from: IEFVHC or IEFVHCB via branch.

Exit to: When IEFVHEB is used as a subroutine, it returns to its caller via branch; when it has been used to initialize a JMR, or to process a SYSCHK DD statement or a PROC statement, it branches to IEFVEA.

IEFVHF — Converter Termination Routine

Operation: This module frees storage space used by the converter, issues a terminate request for the scheduler JCL facility services, issues messages to the operator

about the status of the job after converter processing, initializes SYSCHK DD statement processing and deactivates the converter ESTAE environment.

Entry from: IEFVFA via branch.

Exit to: IEFVHA when continuation of JCL text is expected; IEFVHCB when no continuation is expected or to continue SYSCHK DD processing; IEFVHEB if a procedure statement is being overridden.

IEFVHH — Interpreter ENQUEUE Routine

Operation: This module writes job and step tables into SWA as required and then performs miscellaneous clean-up processing for the interpreter.

Entry from: IEFVHE via branch.

Exit to: IEFVHN via branch.

IEFVHL — Converter NULL Statement or END-OF-FILE Processor

Operation: This routine performs end-of-file processing for JCL statements in the JCL data set or in a cataloged or in-stream procedure.

Entry from: IEFVHEB via branch.

Exit to: IEFVHA, via branch, to continue reading a procedure; IEFVHEB, via branch to continue processing a JCL statement from a procedure; IEFVHCB, via branch, to continue reading JCL statements from the JCL data set; IEFVHF, via branch, when both the JCL data set and all procedures are completed.

IEFVHM — Converter Command Verb Validation Routine

Operation: This routine determines if a valid command has been specified on a JCL statement.

Entry from: IEFVHCB via branch.

Exit to: IEFVHA via branch.

IEFVHN — Interpreter Termination Routine

Operation: This module frees storage space used by the interpreter, issues a terminate request for the scheduler JCL facility services, issues messages to the operator about job status at the end of interpreter processing, and deactivates the ESTAE environment over the interpreter.

Entry from: IEFVHH via branch.

Exit to: The interpreter's original caller, SWA create interface, IEFIB600, via branch.

IEFVHQ — Converter SWA Manager Interface

Operation: This routine is the interface between either the converter or interpreter and the SWA manager or between the interpreter and the journal merge routine.

Entry from: IEFVDA, IEFVEA, IEFVFB, IEFVHA, IEFVHH, IEFVINA, or IEFVJA via branch.

Exit to: Caller via branch.

IEFVHR — Converter/Interpreter Operator Message Module

Operation: This module puts converter and interpreter error messages for the operator into the message data set.

Entry from: IEFVHA, IEFVHF, IEFVHM, or IEFVHN via branch.

Exit to: Caller via branch if recovery is possible; to IEFVHN, via branch, if recovery is not possible.

IEFVH1 — Converter Initialization Routine

Operation: This routine performs initialization processing for the converter, obtains storage space for work areas and buffers, and establishes the ESTAE environment over the converter.

Entry from: The job entry subsystem or the master subsystem.

Exit to: IEFVHA via branch.

IEFVINA — Converter In-stream Procedure Control and GTF Routine

Operation: This module reads in-stream procedures from the JCL data set, records each procedure, and saves each one in compressed form in the scheduler work area (SWA).

Entry from: IEFVHCB via branch.

Exit to: IEFVHA, via branch, for each successive statement in an in-stream procedure.

IEFVINB — Converter In-stream Procedure Directory Search Routine

Operation: This routine searches through the procedure directory for a procedure name specified on a JCL statement. When it locates the procedure name, it returns to the calling routine the address at which that procedure begins.

Entry from: IEFVINA via branch.

Exit to: Caller via branch.

IEFVINC — Converter In-stream Procedure Directory Build Routine

Operation: This module builds an entry for a specified procedure in the in-stream procedure directory.

Entry from: IEFVINA via branch.

Exit to: Caller via branch.

IEFVIND — Converter In-stream Procedure Expand Interface Routine

Operation: This module functions as an access method for IEFVHA. Whenever IEFVHA issues a READ macro instruction for a record from an in-stream procedure, IEFVIND sets up required parameter lists and then invokes IEZDCODE to expand the record.

Entry from: IEFVHA via branch.

Exit to: IEZDCODE, via branch, to expand a record; to IEFVHQ, via branch, to read another record; to caller, via branch, when processing has completed.

IEFVINE — Converter In-stream Procedure Syntax Checker

Operation: This routine checks the label field of a PROC or PEND statement appearing in an in-stream procedure.

Entry from: IEFVINA via branch.

Exit to: Caller via branch.

IEFVJA — Interpreter JOB Statement Processor

Operation: This module creates and initializes the control blocks and tables required by a JOB statement and also checks the validity of JOB statement keyword values.

Entry from: IEFVHE via branch.

Exit to: Caller via branch.

IEFVJDTI — JDT-Defined JCL Interpreter

Operation: This module interprets the JDT-defined JCL internal text and builds the associated scheduler work blocks (SWBs).

Entry from: IEFVGK.

Exit to: Caller.

IEFVJDTV — JDT-Defined JCL Verb Processor

Operation: This module processes the JDT-defined verb statements.

Entry From: IEFVHE.

Exit To: Caller.

IEFVKMSG — Test EXEC Dependency Codes Message

Operation: This module contains the message text of the message issued by IEFBB402, if a step is not run because of the COND parameter on the EXEC statement.

Entry from: Not applicable (non-executable module).

Exit to: Not applicable (non-executable module).

Called routines: Not applicable (non-executable module).

IEFXB500 — Journal Write Routine

Operation: This module writes critical control blocks to the job journal for purposes of restart or termination.

Entry from: SWA manager, initiator, VIO, and the scheduler JCL facility.

Exit to: Caller.

IEFXB601 — Journal Merge Routine

Operation: This module merges the control blocks from the job journal to the scheduler work area (SWA) to establish a pre-job failure-environment.

Entry from: IEFIB605.

Exit to: IEFIB605.

IEFXB602 — Move Mode Restart Interface Routine

Operation: This module builds a virtual address table (VAT) to be used when the journal merge routine reconstructs SWA.

Entry from: IEFVHQ.

Exit to: IEFVHQ.

IEFXB603 — Automatic/Checkpoint Restart Message Module

Operation: This module contains message texts for journal service and checkpoint routines. The module is non-executable.

Entry from: IEFXB500, IEFRPREP, IEFXB609, IEFXB601.

Exit to: Caller.

IEFXB604 — Step Header Create

Operation: Except for automatic step or automatic checkpoint restarts, this module builds the step header record for each job step, before putting anything else in the job journal for the step.

Entry from: IEFSD162.

Exit to: IEFSD162.

IEFXB609 — Data Set Descriptor Record Processor

Operation: This module makes the SWA entries for restarting jobs reflect the job environment at the time a checkpoint was issued.

Entry from: IEFIB605.

Exit to: IEFIB605.

IEFXB610 — Open Interface Module

Operation: This routine opens the checkpoint data set.

Entry from: IEFXB609.

Exit to: IEFXB609.

IEFXB611 — Locate Mode Restart Interface Processing

Operation: This module builds a virtual address table (VAT) for each SWB to be used when the journal merge routine reconstructs the SWA.

Entry from: IEFSJWRT.

Exit to: IEFSJWRT.

IEFXVNSL — User-Replaceable Non-Standard Label Routine

Operation: This user-replaceable module returns a code of 4 if it is called because no user non-standard label handling routine exists to process an NSL label volume.

Entry from: IEFAB473.

Exit to: Caller.

Called routines: None.

IEWSUOVR — Overlay Supervisor Resident Module

Operation: This module performs overlay requests.

Entry from: Overlay segment.

Exit to: Overlay supervisor processor IEWSWOVR.

IEWSWOVR — Overlay Supervisor Processor

Operation: Performs the overlay request and causes segments to be brought into virtual storage.

Entry from: Overlay supervisor resident module IEWSUOVR.

Exit to: Overlay segment.

IEZDCODE — Interpreter In-stream Procedure Decompression Subroutine

Operation: This module re-expands the in-stream procedure records that were compressed by IEZNCODE.

Entry from: IEFVIND via branch.

Exit to: Caller via branch.

IEZNCODE — Interpreter In-stream Procedure Compression Subroutine

Operation: This module compresses the data records that comprise an in-stream procedure by replacing blanks with count fields. It then blocks the compressed records.

Entry from: IEFVINA via branch.

Exit to: Caller via branch.

IFAEASI — SMF System Address Space Processing

Operation: This module, which is scheduled as an SRB, records SMF accounting data for system address spaces that do not go through full-function start.

Entry from: SRB scheduler, scheduled by IFAINIT; IFAEASIL.

Exit to: Caller.

Called routines: IEEMB839 (SMF timer enqueue).

IFAEASIL — SMF System Address Space Setup

Operation: This module schedules an SRB to set up the SMF timer elements for address spaces that do not go through full-function start and are started after SMF initialization is complete.

Entry from: IEEPRWI2.

Exit to: Caller.

Called routines: IFAEASI, IEEMB839.

IFAINIT — SMF System Address Space Processing Initialization

Operation: This module is called during SMF initialization. It checks all address spaces and schedules an SRB to the system address spaces that have not gone through full function start. The SRB causes SMF accounting data to be recorded for the address space.

Entry from: IEEMB823.

Exit to: Caller.

Called routines: IFAEASI, IEEMB839.

IFARPORT — SMF Summary Activity Report for IFASMFDP

Operation: This module writes a summary report of records processed by IFASMFDP (the SMF dump program).

Entry from: IFASMFDP.

Exit to: Caller.

IFASMFDP — SMF Data Set Dump Module

Operation: This module parses input, supplies defaults, and lists resultant options. It dumps input data sets in response to the keyword DUMP and clears VSAM input data sets in response to keyword CLEAR.

Entry from: Problem program.

Exit to: Caller.

Called routines: IEEMB832, IEEMB833, IEEMB824, IFARPORT via BALR.

IGC0203E (IEEJB840) — Write-to-Programmer

Operation: Process write-to-programmer requests (messages) for WTO and WTOR macro instructions with routing code 11.

Entry from: IEAVVWTO.

Exit to: IEAVVWTO.

IGC008E — NSLREPOS Interface

Operation: This module interfaces with the user written routine, NSLREPOS, to verify NSL tapes and to determine if re-positioning of the tape is required.

Entry from: Supervisor call instruction (SVC) interrupt handler.

Secondary entry point: IGC0408E (return from user written routine, NSLREPOS).

Exit to: Caller.

Called routines: NSLREPOS (user written).

IGE0660A — Error Recovery Procedures (ERP) Interface

Operation: After a permanent disk or I/O error occurs, this module tests the existing conditions to determine whether DDR can be used to correct the error. If so, it schedules the DDR task.

Entry from: Outboard recorder (OBR).

Exit to: Dispatcher.

Called routines: IGFDE0, IGFDV1.

IGFCCHCR — CCH Central Routine

Operation: This module creates error recovery procedure information blocks (ERPIBs) containing channel dependent information for the error recovery procedures (ERPs). IGFCCHCR allows an attempted recovery that would otherwise be a permanent I/O error. It creates a record for each channel error detected so that it may be recorded on SYS1.LOGREC.

For entry IGFCCHAS: This routine formats and issues the CCH operator awareness message. It puts out the CCH LOGREC record and the message asynchronously using the RECORD macro of RTM.

For entry IGFCCHEX: This routine is the common exit from CCH, it determines whether the exit is to I/O supervisor (IOS) or to I/O restart.

For entry IGFCCHFE: This routine completes the work ERPIB, determines the channel type from the channel availability table (CAT) and calls the appropriate channel dependent analysis routine.

For entry IGFCCHFR: This routine receives control from RTM to perform recovery from a program check by building the register update block.

For entry IGFCCHIO: This routine handles errors for which CCH was unable to create the necessary error recovery procedure information block (ERPIB). It schedules a retry by the ERPs, or posts the error as permanent.

For entry IGFCCHMP: This routine fills in the multiprocessing portion of the CCH record.

For entry IGFCCHRD: This routine creates the channel error record to be recorded on SYS1.LOGREC.

For entry IGFCCHRV: This routine determines if the channel reconfiguration hardware (CRH) is active for the channel with the detected error.

For entry IGFCCHUC: This routine searches the unit control block (UCB) for active devices on the channel indicating a channel failure.

Entry from: IOS (IECIOSCN).

Exit to: IOS.

Called routines: IGFCIC, IGFC60, IGFC70, IGFC80, IGFPSAD0, IGFPSAE0.

IGFCCHED — External Damage Type Machine Check Processor

Operation: This module processes external damage type machine checks for the channel check handler. It analyzes all available channels as indicated in the channel availability table (CAT), and schedules the proper routine based on whether a lost channel condition occurred.

Entry from: IGFPMSCA.

Exit to: Caller.

Called routines: Schedules IECVIRST or IECVRSTI, if lost channel(s) detected.

IGFCIC — Integrated Channel Analysis Routine

Operation: This module performs the dependent analysis for the integrated channels. It checks the validity of the limited channel logout (LCL) and sets indicators in the error recovery procedure information block (ERPIB). It also checks the I/O extended logout and saves the length and address for processing by the record entry build routine (IGFCCHRE).

Entry from: IGFCCHCR.

Exit to: Caller.

Called routines: None.

IGFC60 — 2860 Channel Analysis Routine

Operation: This module performs the channel dependent analysis for the 2860 channel. It sets indicators for validity bits, error conditions, termination codes, and sequence codes in the work error recovery procedure information block (ERPIB).

Entry from: IGFCCHCR.

Exit to: Caller.

Called routines: None.

IGFC70 — 2870 Channel Analysis Routine

Operation: This module performs the channel dependent analysis for the 2870 channel. It sets indicators for validity bits, error conditions, termination codes, and sequence codes in the work error recovery procedure information block (ERPIB).

Entry from: IGFCCHCR.

Exit to: Caller.

Called routines: None.

IGFC80 — 2880 Channel Analysis Routine

Operation: This module performs the channel dependent analysis for the 2880 channel. It sets indicators for validity bits, error conditions, termination codes, and sequence codes in the work error recovery procedure information block (ERPIB).

Entry from: IGFCCHCR.

Exit to: Caller.

Called routines: None.

IGFDD0 — DDR Disk Select

Operation: This module processes a swap request for a disk device.

Entry from: IGFDI0.

Exit to: Caller.

Called routines: IGFDE0, IGFDD1, IGFDV0, IGFDM0.

IGFDD1 — DDR Disk Validate

Operation: This module validates the disk devices involved in a disk swap.

Entry from: IGFDD0, IGE0660A, IGFDV0.

Exit to: Caller.

Called routines: IGFDV1, IGFDE0.

IGFDE0 — DDR Extended Specify Task Abnormal Exit (ESTAE) Controller

Operation: This module establishes or removes the error recovery environment for DDR.

Entry from: IGF2503D, IGFDD0, IGFDV0, IGE0660A, IGFDT0, IGFDS1, IGFDU0, IGFDI0, IGFDI1.

Exit to: Caller.

Called routines: None.

IGFDE1 — Extended Specify Task Abnormal Exit (ESTAE) Handler

Operation: This module releases the dynamically allocated storage and allocated resources of a module that has experienced an error. It also determines for the recovery termination manager whether a retry attempt should be made.

Entry from: Recovery termination manager.

Exit to: Caller.

Called routines: IGFPxxS, if this special exit routine exists.

IGFDI0 — DDR Initiator

Operation: This module controls processing of the DDR queues. It locates a request for device swap, calls the appropriate routine to handle it, removes it from the queue, and repeats this process until the queue is empty. IGFDI0 also controls the repositioning of a tape volume for a tape swap.

Entry from: Dispatcher, attached by IGFDS1.

Exit to: Caller.

Called routines: IGFDU0, IGFDD0, IGFDT0, IGFDI1, IGFDE0, IGFDT2.

IGFDI1 — DDR Termination Module

Operation: This module terminates processing of individual DDR requests and removes the DDR control block (DDRCOM) from the queue. It may also terminate the entire queue.

Entry from: IGFDI0 or IGFDE0.

Exit to: Caller.

Called routines: IGFDE0, IGFDM0, IGC015.

IGFDL1 — Candidate Lookup

Operation: This module locates a compatible, on-line, unallocated device to be used in a system-initiated device swap.

Entry from: IGFDV0.

Exit to: Caller.

Called routines: IGFDT1, IGFDD1, IGFDE0.

IGFDMTXT — DDR Message Text Module

Operation: This module contains the text segments used to build all messages issued by the DDR message writer (IGFDM0).

Secondary entry point: IGFDMIND.

Secondary entry point: IGFDTIND.

Secondary entry point: IGFDIIND.

Secondary entry point: IGFDHIND.

Called routines: None.

IGFDM0 — DDR Message Writer Routine

Operation: This module issues a DDR message to inform the operator of the status of DDR, to prompt the operator for information necessary for the swap to continue, or to inform the operator of the status of the swap.

Entry from: IGFDD0, IGFDT0, IGFDU0, IGFDV0.

Exit to: Caller.

Called routines: IGFDE0.

IGFDM1 — DDR I/O Module

Operation: This module performs all DDR I/O, except writing console messages.

Entry from: IGFDV0.

Exit to: Caller.

Called routines: IGFDE0.

IGFDR0 — DDR Recorder

Operation: This module formats DDR LOGREC records and requests LOGREC recording.

Entry from: IGFDV0.

Exit to: Caller.

Called routines: IGFDE0, IGFDM1.

IGFDS0 — DDR Service Request Block (SRB) Routine

Operation: This module creates the DDR task.

Entry from: Dispatcher, SRB scheduled by IGF2503D, IGE0660A, IGFDI0.

Secondary entry point: DSOFRR, entered by the recovery termination management after abnormal termination occurs while IGFDS0 is executing.

Exit to: Caller.

Called routines: Stage 1 exit effector, stage 2 exit effector, IGC015.

IGFDS1 — DDR Attacher

Operation: This module issues an ATTACH macro instruction to give control to IGFDI0, the DDR initiator.

Entry from: Dispatcher, created (IRB) by IGFDS0.

Exit to: Caller.

Called routines: IGFDE0, IGFDI1 (as ESTAE exit).

IGFDT0 — DDR Tape Select Routine

Operation: This module processes a swap request for a magnetic tape device.

Entry from: IGFDI0.

Exit to: Caller.

Called routines: IGFDT1, IGFDV0, IGFDR0, IGFDM0, IGFDE0, IGFDV1.

IGFDT1 — DDR Tape Validate Routine

Operation: This module validates the tape device involved in a tape swap.

Entry from: IGE0660A, IGFDT0, IGFDV0.

Exit to: Caller.

Called routines: IGFDE0.

IGFDU0 — DDR Unit Record Select

Operation: This module processes a swap request for a unit record device.

Entry from: IGFDI0.

Exit to: Caller.

Called routines: IGFDV0, IGFDE0, IGFDM0.

IGFDU1 — DDR Unit Record Validate

Operation: This module validates the unit record devices involved in a unit record swap.

Entry from: IGFDU0, IGFDV0.

Exit to: Caller.

Called routines: IGFDV1, IGFDE0.

IGFDV0 — DDR Common Select Routine

Operation: This module locates an alternate device to replace the device on which a permanent error occurred, and then performs the device swap for DDR.

Entry from: IGFDD0, IGFDT0, IGFDU0.

Exit to: Caller.

Called routines: IGFDE0, IGFDM0, IGFDL1, IGFDR0, IGFDM1, IGFDD1, IGFDT1, IGFDU1, IGFDW0.

IGFDV1 — DDR Device Independent Validate

Operation: This module determines whether the devices involved in the DDR swap are valid and compatible, and whether IOS has purged the user's swap request.

Entry from: IGFDT0, IGFDD1, IGFDU1.

Exit to: Caller.

Called routines: IGFDE0, IEEVDEV.

IGFDW0 — DDR Unit Control Block (UCB) Swap

Operation: This module swaps the physical unit control blocks of the failing and replacement devices by invoking the IOSGEN function of IOS.

Entry from: IGFDV0.

Exit to: Caller.

Called routines: IGFDE0, IGFDR0, IGFDM1, IOSGEN.

IGFPEXIT — Machine Check Handler Exit Routine

Operation: This module performs processor-related initialization functions when entered from IGFPINIT at system initialization time. It performs threshold analysis by interrogating the various machine-check threshold blocks when called by IGFPMRTM.

Entry from: IGFPMRTM, IGFPBUCR, IGFPEXI2.

Exit to: Caller.

Called routines: IGF01MMM.

IGFPKREF — Storage Key Refresh Routine

Operation: This module provides an interface which enables a storage key to be refreshed. It allows continued system execution when a machine check occurs due to an uncorrected storage key error.

Entry from: IEAVRCF.

Exit to: Caller.

Called routines: IGFPMFRS.

IGFPMCIH — Machine Check Interrupt Handler

Operation: This module performs machine-check recursion analysis, saves the error environment, and routes control to the various error analysis subroutines.

Entry from: Loading the machine check new PSW.

Secondary entry point: IGFPMCE2: The model-dependent support routine, IGF00MMM, returns to this entry point.

Secondary entry point: IGFPMCPC: At this entry point the module intercepts program checks that occur while IGFPMCIH is executing.

Secondary entry point: IGFPMCRI: The module uses this entry point when a restart interruption occurs while IGFPMCIH is executing.

Exit to: IGFPTERM for processor termination, IGFPMRTM to continue machine check processing.

Called routines: IGF00MMM, IGFPMHCA, IGFPMPFX, IGFPMKTA, IGFPMSCA, ICFBCF00 at entry point ICFBDE00.

IGFPMHCA — Machine Check Handler Hard Machine Check Analysis

Operation: This module analyzes the model-independent fixed logout area to determine if hard machine check conditions are present. If present, it sets indicators in the machine check error indicator area of the LOGREC buffer for subsequent analysis, and schedules message IGF971I or IGF972I.

Entry from: IGFPMCIH.

Exit to: Caller.

Called routines: IGFPMFRS, IGFPMTHA, IGFPMMSG.

IGFPMFRS — Functional Recovery Routine Stacker

Operation: This module allows machine check handler modules to establish and use internal recovery routines in an attempt to recover from a machine check, a program check, or a restart interruption that occurs while they are executing.

Entry from: IGFPMPFX, IGFPMHCA, IGFPMKTA, IGFPMMSG, IGFPMTHA, IGFPMSCA, IGFPKREF.

Secondary entry point: IGFPMSUP: This entry point is used by the machine check handler super functional recovery routine.

Exit to: Caller.

Called routines: IEAVEABD.

IGFPMKTA — Machine Check Clock Timer Analysis

Operation: This module analyzes the model-independent fixed logout area to determine if clock and timer damage conditions are present. If it finds an error in the time-of-day (TOD) clock, clock comparator, processor timer, or interval timer, it sets an indicator in the machine check error indicator area of the LOGREC buffer. If processor timer, clock comparator, or TOD clock damage has occurred, this module calls IGFPMTHA to perform threshold analysis.

Entry from: IGFPMCIH.

Exit to: Caller.

Called routines: IGFPMTHA, IGFPMFRS.

IGFPMMSG — Machine Check Handler Message Scheduler

Operation: This module locates an available message buffer in the message buffer pool, and prepares it for subsequent processing by the machine check asynchronous message writer (IGFPWMSG).

Entry from: IGFPMHCA.

Exit to: Caller.

Called routines: IGFPMFRS.

IGFPMPFX — Prefix Routine

Operation: This module converts an absolute address to a real address for later use by IEAVTRTH, which converts it to a virtual address.

Entry from: IGFPMCIH.

Exit to: Caller.

Called routines: IGFPMFRS.

IGFPMRTM — Machine Check Handler Recovery Termination Management Interface

Operation: This module prepares the error environment data record for subsequent recording on SYS1.LOGREC and links to the recovery termination manager (RTM) to recover the error condition.

Entry from: IGFPMCIH.

Exit to: IGFPEXIT.

Called routines: IEAVTRTH (via CALLRTM).

IGFPMSCA — Machine Check Handler Soft Machine Check Analysis

Operation: This module analyzes the model-independent fixed logout area to determine if soft machine check conditions (system recovery condition, degradation error, or external damage) are present. If so, it sets indicators in the machine check error indicator area of the LOGREC buffer for subsequent analysis. If a system recovery condition, degradation error, or external damage is present, this module calls IGFPMTHA to perform threshold analysis.

Entry from: IGFPMCIH.

Exit to: Caller.

Called routines: IGFPMFRS, IGFPMTHA, IGFCCHED.

IGFPMTHA — Machine Check Threshold Analysis

Operation: This module counts recoverable storage retry and buffer errors on soft machine checks, and compares this count with the threshold values of IGFTHB. When the threshold is reached, it schedules an operator message. IGFPMTHA also analyzes the following hard machine checks: instruction processing damage, system damage, an invalid PSW or register, time-of-day clock damage, processor timer damage, and clock comparator damage. When the threshold is reached, this module informs the caller, which then invokes IGFPTERM to perform processor termination.

Entry from: IGFPMSCA, IGFPMKTA, IGFPMHCA.

Exit to: Caller.

Called routines: IGFPMMSG, IGFPMFRS.

IGFPNRFH — Nucleus Page-in

Operation: This module pages in a single unchanged nucleus page to refresh the area affected by an uncorrected storage error on that nucleus page.

Entry from: Dispatcher.

Exit to: Caller.

Called routines: IEAVFP.

IGFPTAIM — Machine Check Handler Alternate CPU Recovery Interface

Operation: This module initiates alternate CPU recovery (ACR) by giving control to IEAVTACR.

Entry from: IGFPTSIG, IGFPXMFA.

Exit to: IEAVTACR.

IGFPTCON — Machine Check Handler Console Write Routine

Operation: This module informs the operator that the system is to be terminated.

Entry from: IGFPTSIG.

Exit to: Caller.

IGFPTERM — Machine Check Handler Processor Termination

Operation: This module performs either system termination or processor termination. If an unrecoverable machine check occurs, it puts the executing processor in a stop state. If system termination is requested, it puts all online processors in disabled wait states.

Entry from: IGFPMCIH, ICRMSG00, IEAVEPC, IEAVESPR, IEAVTACR, IEAVGM00, IEAVNIPM.

Exit to: IEEVSTOP, IGFPTSIG.

Called routines: IGFPTSIG.

IGFPTREC — Machine Check Handler Emergency Recorder

Operation: During system termination, this module attempts to write records in the SYS1.LOGREC data set.

Entry from: IGFPTSIG.

Exit to: Caller.

IGFPTSIG — Emergency Signal External Second Level Interrupt Handler

Operation: During processor termination, this module invokes alternate CPU recovery (ACR). During system termination, it invokes the emergency recorder and console writer routines, and puts the system in a disabled wait state.

Entry from:

IGFPTERM for processor or system termination initiated by the machine check handler,

IGFPXMFA when a malfunction alert is received,

IEAVEES when an emergency signal is received.

Exit to: IGFPTAIM, IEAVEEXT,

Called routines: IGFPTAIM, IGFPSAD0, IGFPSAE0, IGFPTREC, IGFPTCON.

IGFPXMFA — Malfunction Alert External Second Level Interrupt Handler

Operation: This module handles the malfunction alert external interrupt received from the failing processor.

Entry from: IEAVEEXT.

Exit to: Caller.

Called routines: IGFPTSIG, IGFPTAIM.

IGFPWMSG — Machine Check Handler Asynchronous Message Writer

Operation: This module writes an operator message scheduled by the mainline machine check handler. It executes asynchronously to the mainline machine check handler and operates under the master scheduler task.

Entry from: IEAVTRET.

Exit to: Caller.

IGFTMCHK — Missing Interruption Handler

Operation: This module establishes the disabled interruption exit (DIE) routine, which gains control at specified intervals to search for missing MIH conditions. IGFTMCHK records missing I/O interruptions on the SYS1.LOGREC data set, and informs the operator of all missing MIH conditions.

Entry from: The dispatcher (IEAVEDS0).

Exit to: The dispatcher (IEAVEDS0).

IGFTMC00 — Missing Interruption Handler DIE Routine

Operation: This module schedules an SRB which has an entry point (IGFTMC01) within this module. When entered at IGFTMC01, this module scans the UCBs for missing MIH conditions and informs IGFTMCHK when the scan is completed.

Entry from: The timer second level interrupt handler (SLIH).

Secondary entry point: IGFTMC01 via an SRB.

Called routines: Special exit routines, CVT0PT01 and CVTQTE00.

Exit to: The SLIH when entered at IGFTMC00, the dispatcher when entered at IGFTMC01.

IGFTMC01 — See IGFTMC00

IGF2503D — SWAP Command Processor

Operation: This module processes all MODE commands.

Entry from: IEE0403D.

Exit to: IEE0003D, IEE0503D.

IGF2603D — MODE Command Processor

Operation: This module processes all MODE commands.

Entry from: IEE0403D.

Exit to: Caller, IEE0503D.

IGX00013 — MFSTART Mainline Processor

Operation: Controls the initialization and termination of routines that perform RMF functions.

Entry from: IGC0010I.

Exit to: IGC0010I.

IGX00014 — MFDATA SVC Mainline Processor

Operation: Controls the operation of measurement gathering routines that operate once each interval.

Entry from: IGC0010I.

Exit to: IGC0010I.

IGX00025 — TSO Command Counter Module

Operation: This module accumulates, into the type 32 record, data pertaining to specified TSO commands. The data collected is:

- Number of times the command is used
- TCB time for the command
- SRB time for the command
- TPUTs for the command
- EXCPS for the command
- Transactions for the command

Entry from: SVC FLIH.

Exit to: Caller.

IKJEES20 — SEND Command Message Module

Operation: This module contains the messages for the SEND command processor.

Entry from: IEEVSEND, IKJEES10, IKJEES40.

Exit to: Caller.

IKJEFLF — System-Initiated Cancel Schedule Routine

Operation: This module schedules an SRB routine and checks the validity of the input to this routine.

Entry from: IEE3703D, TIOC routines.

Exit to: Caller.

IKJL4T00 — System-Initiated Cancel SRB Routine

Operation: This module synchronizes events between TIOC, LOGON scheduler and its subtasks to provide orderly cancellation of a TSO user.

Entry from: SRB dispatcher.

Exit to: SRB dispatcher.

IKJ5803D — SEND Command Scanner

Operation: This routine scans the SEND command parameter.

Entry from: IEE0403D.

Exit to: IEE0803D, IEE0503D.

ILRACT — Activate VIO ASPCT Routine

Operation: This routine builds a parameter list and calls ILRVSAMI to retrieve the saved VIO ASPCT from SYS1.STGINDEX. The new LGN is stored with the retrieved ASPCT base and the number of slots required to back this VIO data set is recalculated.

Entry from: ILRGOS.

Exit to: ILRGOS.

ILRCMP — ASM I/O Completion

Operation: This module consists of six major routines: the disabled interrupt exit (ILRCMPDI), normal channel end appendage (ILRCMPNE), abnormal channel end appendage (ILRCMPAE), termination (ILRCMP), PCI exit (ILRCMPCI), and notification exit (ILRMCMPNT).

These routines process the individual requests that are represented by the PCCW chained off the IORB. If the I/O is successful the AIAs are returned to page completion and the PCCWs freed. Bad slots will be recorded in a special buffer in SQA.

If the error occurred before the end of the PCCW chain was reached, the remaining PCCWs will be retried.

Entry from:

For entry ILRCMPDI: IECIOSCN.

For entry ILRCMPAE: IECVPST (Post Status).

For entry ILRCMPNE: IECVPST.

For entry ILRCMP: IECVPST or IEAVEDS0, for SRB scheduled by ILRCMP01.

For entry ILRCMPCI: IECIOSCN.

For entry ILRCMPNT: IECIOSCN.

Exit to:

For entry ILRCMPDI: IECIOSCN.

For entry ILRCMPAE: IECVPST.

For entry ILRCMPNE: IECVPST.

For entry ILRCMP: IEAVEDS0.

For entry ILRCMPCI: IECIOSCN.

For entry ILRCMPNT: IECIOSCN.

ILRCMP01 — I/O Completion Recovery Routine

Operation: This recovery routine will attempt to clean up whatever resources have been checkpointed in the ATA and force reprocessing for any requests not yet attempted.

Entry from: IEAVTRTS (RTM).

Exit to: IEAVTRTS.

ILRCMSRB — SRB Routine to Invoke ILRPAGCM

Operation: This routine (running as an SRB) is scheduled by the ASM functional recovery routines (ILRCMP01, ILRDRV01, ILRIOFRR, and ILRSRB01) to invoke ILRPAGCM. RTM restrictions prohibit the ASM functional recovery routines from directly calling ILRPAGCM.

Entry from: IEAVEDS0.

Exit to: IEAVEDS0.

ILRCPBLD — Channel Program Build Routine

Operation: This routine builds a channel program, then starts the I/O operation with:

- RESUME if the channel program is suspended
- Modify the currently running channel program to TIC to the new one just built
- STARTIO if the channel program has ended

Entry from: ILRIODRV.

Exit to: ILRIODRV.

ILRDRV01 — Recovery Routine for ILRIODRV and ILRCPBLD

Operation: This routine eliminates invalid or loop causing control blocks. It restores or reconstructs the environment so that the remaining I/O requests will be properly processed by ILRIODRV and ILRCPBLD.

Entry from: IEAVTRTS (RTM).

Exit to: IEAVTRTS.

ILRFMT00 — Format Routine for ASM

Operation: This routine formats ASM and shared RSM control blocks contained in storage areas passed by the system dump-printing routine (AMDPRDMP).

Entry from: AMDPRUIM.

Exit to: Caller.

ILRFMTCV — Common and VIO Areas Format Routine for ASM

Operation: This routine has three entry points. ILRFMTC formats the page and external page tables. ILRFMTH formats RSMHD and its SPCT, ASMHD and its AIA chain, the private area page tables and external page tables. ILRFMTV formats the LGVT and its associated control blocks.

Entry from: ILRFMT00.

Exit to: ILRFMT00.

ILRFMTPG — Paging Format Routine for ASM

Operation: This routine formats the PART, PCTs, PATs, and all other ASM paging-related control blocks.

Entry from: ILRFMT00.

Exit to: ILRFMT00.

ILRFMTSW — Swap Format Routine for ASM

Operation: This routine formats the SART, SPCT, SATs, and all other ASM swap-related control blocks.

Entry from: ILRFMT00.

Exit to: ILRFMT00.

ILRFRR01 — ASM Recovery Service Routines

Operation: This module contains service routines common to ASM recovery routines. There are three types of service routines: 1) queue verification routines that verify and correct ASM queues, 2) control block verification routines that verify ASM-related control blocks, and 3) ASM's resource manager termination routine (parameter for the PURGEDQ macro).

Entries:

ILRPSRMT — Reschedule an SRB for ILRCMSRB, ILRSIO, ILREDRV, or ILRSWPDR.

ILRVACE — To verify a potential ACE.

ILRVACEQ — To verify a queue of ACEs.

ILRVACQ2 — To verify and correct a queue of ACEs.

ILRVAIA — To verify a potential AIA.

ILRVAIAC — To verify a potential AIA/ACE.

ILRVIORB — To verify a potential IORB-IOSB-SRB-SRB chain.

ILRVLGE — To verify a potential LGE.

ILRVLPRQ — To verify LGE process queue (LGEPROCQ).

ILRVPCB — To verify a potential PCB.

ILRVPCBQ — To verify queue of PCBs (RSMLIOQ).

ILRVPCCW — To verify a potential PCCW.

ILRVPCWQ — To verify a queue of PCCWs.

ILRVSCCW — To verify a potential SCCW.

ILRVSCWQ — To verify a queue of SCCWs.

ILRVSPAQ — To verify a queue of swap AIAs.

ILRVSWTQ — To verify swap wait queue (SARWAITQ).

Entry from:

For all entries except ILRPSRMT: ILRGOS01, ILRCMP01, ILRSRB01, ILRDRV01, ILRSWP01, ILRTERMR, ILRTMI01, ILRIOFRR.

For ILRPSRMT: IEAVEPDQ.

Exit to:

For all entries except ILRPSRMT: Caller.

For ILRPSRMT: IEAVEPDQ.

ILRFRSLT — Free Slot Routine

Operation: This routine frees slots or swap sets, as required by the caller, by resetting the appropriate PAT bit or SAT bit.

Entries:

ILRFRSLT — used by RSM to free slots.

ILRFRSL1 — used by ASM to free slots.

ILRFRSW1 — used by ASM to free swap sets.

Entry from:

For entry ILRFRSLT: IEAVRELS, IEAVAMSI (RSM routines).

For entry ILRFRSL1: ILRIODRV, ILRDRV01, ILRSAV, ILRTERMR, ILRTMRLG, ILRPOS.

For entry ILRFRSW1: ILRPAGCM, ILRSWAP, ILRTERMR.

Exit to: Caller.

ILRGOS — Group Operation Starter

Operation: This module accepts the following group requests: Assign LGN, Save LG/LGN, Activate LG, and Release LG. Each request is attempted to be started immediately. If not started, it is queued for later processing.

Secondary entry point: (ILRFRELG): Dequeues and frees a LGE.

Entry from:

For entry ILRGOS: Virtual Block Processor (VBP).

For entry ILRFRELG: ILRGOS01 or ILRRLG.

Exit to: Caller.

ILRGOS01 — Group Operators Recovery Routine

Operation: This module serves as an ESTAE for save and activate requests and an FRR for release logical group and assign requests. It will only retry for record-only abends. For other errors, the resources will be freed and the error percolated to the recovery routine of the caller of ILRGOS.

Entries:

ILRGOS01: Handle the release or assign errors.

ILRCGOSE: Handle the save or activate errors.

Entry from:

For entry ILRGOS01: IEAVTRTS (RTM).

For entry ILRCGOSE: IEAVTAS1 (RTM).

Exit to: Caller.

ILRIODRV — ASM I/O Driver

Operation: This module contains several entry points used by RSM and ASM to manipulate page data sets.

Entry point ILRIODRV: General frame allocation (IEAVGFA) and ASM swap control (ILRSWAP) call this routine to initiate I/O operations for page data sets.

Entry point ILREDRV: ASM I/O completion (ILRPAGCM) schedules this entry as an SRB to retry requests that were incorrectly completed or that could not be started because of insufficient PCCWs.

Entry point ILRSWLIO: If LSQA swap out (ILRSLSQA) was called by page completion (ILRPAGCM), it schedules this entry as an SRB to start LSQA swapping.

Entry point ILRSWAPO: If LSQA swap out was not invoked by page completion, it calls this entry point directly to start LSQA swapping.

Entry point ILRSWPIN: ASM swap control (ILRSWAP) calls this entry point to start stage 1 swap-in processing.

Entry point ILRVIODR: Page operations starter (ILRPOS) calls this entry point to start VIO paging requests that had been queued behind other work.

Entry from:

For entry ILRIODRV: IEAVGFA, ILRSWAP.

For entry ILREDRV: IEAVEDS0 (scheduled by ILRPAGCM).

For entry ILRSWLIO: IEAVEDS0 (scheduled by ILRSLSQA).

For entry ILRSWAPO: ILRSLSQA.

For entry ILRSWPIN: ILRSWAP.

For entry ILRVIODR: ILRPOS.

Exit to: Caller.

ILRIOFRR — ASM I/O Control Recovery Routine

Operation:

Entry ILRIOFRR: This routine is called by RTM anytime an error occurs during ASM's swap processing, initial page processing, and page completion processing. The mainline of the routine is a router to the appropriate recovery routine or subroutine representing the ASM function in control at the time of the error.

Entry ILRPOS01: This routine attempts recovery for ILRPOS.

Entry from: IEAVTRTS (RTM).

Exit to: IEAVTRTS.

ILRJTERM — ASM Job Termination Resource Manager

Operation:

Entry ILRJTERM: This routine will schedule asynchronous operations to deactivate any VIO data sets still active at job deletion time. The LGE queue is searched to locate these VIO data sets.

Entry ILRJTM01: This routine is the recovery routine for ILRJTERM.

Entry from:

For entry ILRJTERM: IEFSD166 (initiator job deletion module).

For entry ILRJTM01: IEAVTRTS (RTM).

Exit to: Caller.

ILRMSG00 — ASM Message Module

Operation: Entries ILRMSG00 and ILRMSGSP:

This routine writes messages concerning the status of page and swap data sets. It will terminate the system if the condition of these data sets makes it impossible for ASM to continue.

Entry ILRMSG01: This routine is the recovery routine for the termination subroutine of ILRMSG00.

Entry from:

For entry ILRMSG00: ILRCMP, ILRCMP01, ILRDRV01, ILRIODRV, ILRSWP01.

For entry ILRMSGSP: ILRTMI00 (an ASM initialization routine).

For entry ILRMSG01: IEAVTRTS (RTM).

Exit to:

For entries ILRMSG00 and ILRMSGSP: Caller, if the system is not to terminate. IGFPTERM if the system is to terminate.

For entry ILRMSG01: IEAVTRTS.

ILROPS00 — ASM Page or Swap Open Routine

Operation: This routine is entered to locate, mount, and build I/O control blocks for a page or swap data set. At NIP time control blocks may be requested not to be built. After NIP time control blocks are always built.

Entry from: ILRASRIM or ILRPGEXP.

Exit to: Caller.

ILRPAGCM — ASM Page I/O Completion

Operation: This routine analyzes an input chain of AIAs, separates them, and places them onto internal queues for the page and VIO subroutine or the swap subroutine. Each subroutine processes its chain of AIAs. Successful AIAs are returned to RSM.Unsuccessful AIAs may be redriven.

Entry from: ILRCMP or ILRCMSRB.

Exit to: ILRCMP or ILRCMSRB.

ILRPEX — ASM Pool Extender

Operation: This routine attempts to extend the ASM virtual storage pool indicated (ACE, BWK or SWK). ASMVT pool controller information is used and updated.

Entry from: Any ASM module using ILRGMA macro.

Exit to: Caller.

ILRPGEXP — Page and Swap Expansion Module

Operation: This module dynamically adds page and swap data sets to the system when the PAGEADD operator command is issued.

Entry ESTAER is the ESTAE routine for ILRPGEXP.

Entry from:

For entry ILRPGEXP: Attached by IEEVWAIT.

For entry ESTAER: IEAVTAS1 (RTM).

Exit to: Caller.

ILRPOS — Page Operations Starter

Operation:

Entry ILRPOS: This routine receives a string of I/O requests or a single transfer page request. The proper page processing subroutine is called. Requests that cannot be started immediately are queued for later processing.

Entry ILRESTRT: This routine starts AIAs (I/O requests) that are on the LGE process queue.

Entry ILRTRANS: This routine processes transfer page ACEs.

Entry ILRTRPAG: This routine creates a transfer page ACE from the transfer page request (ACA).

Entry from:

For entry ILRPOS: ILRIODRV or ILRTRPAG.

For entry ILRESTRT: ILRSRBC.

For entry ILRTRANS: ILRSRBC or internally by ILRPOS.

For entry ILRTRPAG: IEAVAMSI (RSM).

Exit to: Caller.

ILRPREAD — ASM-Special I/O Driver Routine

Operation: This routine is an I/O driver for ASM initialization and expansion. It builds channel programs and invokes IOS using the STARTIO macro to do the I/O. It contains normal channel end abnormal channel end appendages (both of which are null routines). It contains an I/O termination routine that posts the mainline when the I/O has completed and which releases the LOCAL lock obtained by IOS.

Callers invoke this routine to build and execute one of the following channel programs:

1. To read slots from the PLPA, common, or duplex page data sets
2. To write slots to the PLPA, common, or duplex page data sets
3. To test the cache of a buffered paging device
4. To reset the cache of a buffered paging device
5. To obtain the address of the cache (control block) array element that corresponds to a specific buffered paging device

This routine reads and writes the slots on the PLPA, common, and duplex page data sets in order to access the TPARTBLE (ILRTPARB), QSRCD and record-0 timestamps. This information is needed to perform quick starts and warm starts.

This routine test and resets the caches and obtains the addresses of cache array elements for ASM. ASM uses these services when page or swap data sets reside on a buffered paging device.

Entry from:

For entry ILRPREAD: ILRPGEXP and ASM initialization modules (ILRASRIM, ILRQSRIT, ILRTMI00).

For entries PREADABN and PREADNRM: IECVPST (Post status).

For entry PREADTRM: IECVPST.

For entry ESTAEXIT: IEAVTAS1.

Exit to: For entry ILRPREAD: Caller.

For entries PREADABN and PREADNRM: IECVPST.

For entry PREADTRM: IEAVEDS0 (Dispatcher).

For entry ESTAEXIT: IEAVTAS1.

ILRRLG — Release Logical Group Operator

Operation: This routine releases auxiliary storage (unsaved slots) and control blocks (active ASPCTs) relating to a logical group.

Entry from: ILRGOS or ILRSRBC.

Exit to: Caller.

ILRSAV — Save Operator

Operation: This routine stores a logical group's ASPCTs in the SYS1.STGINDEX data set and flags the individual LPMEs (slot information) as saved.

Entry from: ILRGOS.

Exit to: ILRGOS.

ILRSRBC — VIO SRB Controller

Operation:

Entry point ILRSRBC: This routine is dispatched in the address space for which a group or page operation is pending. It finds the pending work, determines all work that can be started, and starts it.

Entry point ILRSRBRM: This routine cleans up resources. It frees the SRB scheduled to give ILRSRBC control or resets the schedule flag.

Entry from:

For entry point ILRSRBC: IEAVEDS0 for SRB scheduled by ILRGOS, ILRVIOCM, or ILRJTERM.

For entry point ILRSRBRM: IEAVEPDQ.

Exit to: IEAVEDS0.

ILRSRB01 — Recovery Routine for ILRSRBC

Operation: This routine processes all errors which occur in ASM's SRB controller.

Entry from: IEAVTRTS (RTM).

Exit to: IEAVTRTS.

ILRSWAP — ASM Swap Control Module

Operation:

Entry point ILRSWAP: This routine receives swap requests from RSM and controls the placement or retrieval of the pages on either swap data sets or on page data sets (when no swap data sets are available).

Entry point ILRSLSQA: This routine builds and queues swap channel command workareas (SCCWs) for ILRSWPDR to start.

Entry from:

For entry point ILRSWAP: IEAVPIOI or IEAVSWIN.

For entry point ILRSLSQA: ILRPAGCM or internally from ILRSWAP.

Exit to: Caller.

ILRSWPDR — Swap Driver

Operation: This routine drives IOS via the STARTIO macro for pages being written to or read from a swap data set.

Entry from: IEAVEDS0 for SRB scheduled by ILRSLSQA (entry of ILRSWAP) or ILRCMP.

Exit to: IEAVEDS0.

ILRSWP01 — ASM Swap Recovery Routine

Operation: This routine processes errors which occur in ASM's swapping path. ILRSWP01 entry receives control if there is an error in ILRSWPDR. ILRSCWAP entry receives control if there is an error in ILRSWAP. ILRCSLSQ entry receives control if there is an error in ILRSLSQA (an entry in ILRSWAP).

Entry from: ILRIOFRR.

Exit to: ILRIOFRR.

ILRTERMR — ASM Address Space Termination Routine

Operation:

Entry ILRTERMR: This routine receives control during termination of any address space. All ASM resources for the address space including storage, control blocks, and auxiliary storage slots are freed or marked to be freed when in-process operations complete.

Entry ILRSLTRV: This routine determines if sufficient unreserved slots exist to allow creation of another address space.

Entry TERMRFRR: Recovery routine for ILRTERMR.

Entry from:

For entry ILRTERMR: IEAVTMTC (RTM).

For entry ILRSLTRV: IEAVITAS.

For entry TERMFRR: IEAVTRTS (RTM).

Exit to:

For entry ILRTERMR: IEAVTMTC.

For entry ILRSLTRV: IEAVITAS.

For entry TERMFRR: IEAVTRTS.

ILRTMI01 — Task Mode Initialization Recovery

Operation: This ESTAE routine provides recovery for ILRTMRLG and its calls to ILRTMI00 and ILRVSAMI.

Entry from: IEAVTAS1 (RTM).

Exit to: IEAVTAS1.

ILRTMRLG — Task Mode Release Logical Group

Operation: This routine loads and calls ILRTMI00 to complete ASM initialization. It then waits to be posted for task mode release logical group processing (that is, erase saved ASPCTs and release the slots assigned in those ASPCTs).

Entry from: IEAVEDS0 (when initially attached by IEEMB860 or subsequently posted by ILRRLG or ILRTMI01).

Exit to: IEAVEDS0 via WAIT.

ILRVIOCM — VIO Completion

Operation: This routine stores the newly-assigned LSID in an ASPCT for each complete I/O processing of a VIO page.

Entry from: ILRPAGCM.

Exit to: ILRPAGCM.

ILRVSAMI — VSAM Interface Routine

Operation: This routine interfaces with VSAM to complete any I/O to SYS1.STGINDEX involved in processing save, activate, and release logical group operations of ASM.

Entry from: ILRACT, ILRSAV, or ILRTMRLG.

Exit to: Caller.

IRARMANL — SRM Syntax Analyzer

Operation: This module uses a syntax table (IPSYNTAB, ICSYNTAB or OPSYNTAB) to analyze the syntax of input data. Depending on the analysis, it transfers control to the required option initialization routine.

Entry from: IRARMIPS, IRARMICS, IRARMOPT, and IRARMANL (recursive).

Exit to: Caller.

IRARMCNS — SRM Constants Module

Operation: Provides the pre-assembled tables used within the SRM for non-refreshable data.

Entry from: Not applicable.

Exit to: Not applicable.

IRARMCPM — SRM CPU Management

Operation: Consists of a set of routines that monitor the system-wide processor load. They recommend users for swapping when the system is under- or over-utilized, and users exits that would improve the situation if swapped in or out.

Entry from: IRARMCTL.

Exit to: IRARMCTL.

IRARMCPU — SRM CPU Adjustment Array

Operation: This module contains model-dependent constants used by SRM to adjust some of its parameters.

Entry from: Not applicable (non-executable code).

Exit to: Not applicable (non-executable code).

IRARMCTL — SRM Control Algorithm

Operation: This routine routes control to various SRM routines both as a result of explicit requests, and as a result of the expiration of internally scheduled periodic intervals. The routine defers the routing of control to certain requested routines until concurrently executing SRM processing completes. SRM control includes the routines that analyze the system users to determine which should be swapped in or out.

Entry from: IRARMEVT when a SYSEVENT has been received, from IRARMFIP when a timed algorithm is due to be run, and from the dispatcher for the execution of an SRM SRB.

Exit to: IRARMEVT or IRARMINT.

IRARMERR — SRM Functional Recovery

Operation: This module provides functional error recovery for SRM.

Entry from: Recovery termination manager.

Exit to: Recovery termination manager.

IRARMEVT — SRM SYSEVENT Routers and Processors

Operation: This module translates information describing changes in the status of an address space or in the system environment from an external representation to an internal representation that can be acted upon by SRM. Also performs a limited number of specific services for other components.

Entry from: IRARMINT.

Exit to: IRARMINT or IRARMCTL.

IRARMFIP — SRM Fast Interface Path

Operation: This module performs the processing for those SYSEVENTs whose execution frequency or short path length makes the generalized path of IRARMINT unnecessary.

Entry from: Any issuer of a SYSEVENT pre-defined as a candidate for a fast path event.

Exit to: The SYSEVENT issuer, IRARMCTL, or IRARMINT.

IRARMFMT — SRMDATA Print Dump Format Exit

Operation: If the SRMDATA format control verb is specified in the print dump input, this module formats the three SRM swap queues of user control blocks (OUCBs). They are the WAIT, OUT, and IN queues. It also formats the domain table (DMDT), which controls the system's multiprogramming level (MPL).

Entry from: AMDPRUIM (an AMDPRDMP utility module).

Exit to: AMDPRUIM.

IRARMICS — SRM Installation Control Specification Processor

Operation: This module scans the installation control specification list in the IEAICSxx member of SYS1.PARMLIB and builds a control block for installation control specification information.

Entry from: IEEMB812.

Exit to: IEEMB812.

IRARMINT — SRM Interface Program

Operation: This module performs the necessary processing for most branch entry and SVC SYSEVENTS to permit information pertaining to individual address spaces or system resources to be passed to and retrieved from SRM.

Entry from: Most SYSEVENT issuers (branch type SYSEVENT), SVC processors (SVC type SYSEVENT), IRARMEVT, or IRARMCTL.

Exit to: Most SYSEVENT issuers (branch type SYSEVENT), type 1 SVC exit routine (SVC type SYSEVENT), or the dispatcher (if an SVC entry and an SRB must be scheduled immediately).

IRARMIOM — SRM I/O Management

Operation: This module consists of a set of routines that monitor the I/O logical channel usage of certain address spaces. These routines recommend address spaces for swapping based upon the extent to which the swap-in or swap-out of the address space would correct a detected I/O imbalance.

Entry from: IRARMCTL.

Exit to: IRARMCTL.

IRARMIPM — IPS/Installation Control Specification/OPT Message Module

Operation: This module contains the text for syntax error messages issued by IRARMIPS, IRARMICS, and IRARMOPT.

Entry from: Not applicable.

Exit to: Not applicable.

IRARMIPS — SRM IPS Processor

Operation: This module scans the installation performance specification (IPS) list in the IEAIPSxx member of SYS1.PARMLIB and builds control blocks for IPS information.

Entry from: IEEMB812.

Exit to: IEEMB812.

IRARMMSG — SRM Message Module

Operation: This module contains the WTO list macro forms of SRM messages.

Entry from: Not applicable.

Exit to: Not applicable.

IRARMOPT — SRM OPT Processor

Operation: This module scans the OPT list in the IEAOPTxx member of SYS1.PARMLIB and builds a parameter list containing the new values for the OPT variables.

Entry from: IEEMB812.

Exit to: IEEMB812.

IRARMRMR — SRM Resource Monitor

Operation: This module accumulates several system resource contention indicators, computes the average utilization of these resources and based on the level of utilization, raises or lowers the system multiprogramming level (MPL). It also adjusts system maximum think time for logical swapping.

Entry from: IRARMCTL.

Exit to: IRARMCTL.

IRARMSET — SRM Non-Resident Set to New-Parmlib Member

Operation: This module replaces the internal SRM constants and control blocks with information from the appropriate parmlib member (IEAIPSxx, IEAICSxx, or IEAOPTxx) in response to SET IPS, SET ICS, or SET OPT.

Entry from: IRARMEVT.

Exit to: IRARMEVT.

IRARMSRV — SRM Supervisor Service Request Routine

Operation: This module requests the invocation of specific supervisor services for SRM routines. The supervisor services requested include reordering the ASCB chain, stealing page frames, and obtaining or freeing storage in the SRM storage pool.

Entry from: An SRM routine.

Exit to: The invoking routine.

IRARMSTM — SRM Storage Management

Operation: Consists of a set of routines that control the use of main storage by all address spaces. For non-swappable users, the mechanism of page stealing is used for storage management control; for swappable users, both page stealing and swapping provide the requisite control.

Entry from: IRARMCTL and IRARMEVT.

Exit to: IRARMCTL and IRARMEVT.

IRARMST2 — SRM Storage Management Subroutine Module

Operation: This module consists of a set of routines that evaluate and manage the use of main storage. IRARMST2 is an extension of IRARMSTM.

Entry from: IRARMCTL and IRARMEVT.

Exit to: IRARMCTL and IRARMEVT.

IRARMSWP — Swap Analysis Algorithm

Operation: This module updates the domain descriptor table (DMDT) and selects the users to swap in or out. The swap decisions are based on the current domain targets and user swap recommendation values supplied by the workload manager and the resource load balancers, if they are active.

Entry from: IRARMCTL.

Exit to: IRARMCTL.

IRARMWAR — SRM Workload Activity Recording Routine

Operation: Collects workload activity data when so requested by RMF. Terminates the collection of workload activity data when so requested by RMF, or when the IPS or installation control specification is changed.

Entry from: IRARMEVT or IRARMWLM.

Exit to: IRARMEVT or IRARMWLM.

IRARMWLM — SRM Workload Manager Algorithm Module - Part 1

Operation: Consists of a set of routines that update user transaction statistics, and monitor the workload level at which individual users are receiving service.

Entry from: IRARMCTL and IRARMEVT.

Exit to: IRARMCTL and IRARMEVT.

IRARMWLS — SRM Workload Manager Algorithm Module - Part 2

Operation: Consists of a set of routines that perform user-ready processing and queue processing for subsystem transaction reporting. This module is an extension to IRARMWLM.

Entry from: IRARMCTL and IRARMEVT.

Exit to: IRARMCTL and IRARMEVT.

ISGxxxxx

Modules whose names begin with "ISG" are described in *OS/VS2 Global Resource Serialization Logic.*

Index

OS/VS2 System Logic Library
Volume 11

LY28-1099-1

READER'S COMMENT FORM

This manual is part of a library that serves as a reference source for system 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.***

Possible topics for comment are:

Clarity Accuracy Completeness Organization Coding Retrieval Legibility

If you wish a reply, give your name, company, mailing address, and date:

_____

_____

_____

_____

What is your occupation? _____

How do you use this publication? _____

Number of latest Newsletter associated with this publication: _____

Thank you for your cooperation. No postage stamp necessary if mailed in the U.S.A. (Elsewhere, an IBM office or representative will be happy to forward your comments or you may mail directly to the address in the Edition Notice on the back of the title page.)

Note: Staples can cause problems with automated mail sorting equipment.
Please use pressure sensitive or other gummed tape to seal this form.

Cut or Fold Along Line

LY28-1099-1

Reader's Comment Form

Cut or Fold Along Line

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

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE:

International Business Machines Corporation
Department D58, Building 920-2
PO Box 390
Poughkeepsie, New York 12602

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

IBM®

OS/VS2 System Logic Library Volume 11 (File No. S370-36) Printed in U.S.A. LY28-1099-1

OS/VS2 System Logic Library Volume 11 (File No. S370-36) Printed in U.S.A. LY28-1099-1

LY28-1099-1