LY28-1170-1
File No. S370-36

Program Product

MVS/Extended Architecture Resource Measurement Facility (RMF) Program Logic Manual Volume 1 Part 1

Program Number 5665-274
Version 3, Release 3

IBM

Second Edition (July, 1985)

This edition applies to Version 3 Release 3 of the program product RMF (Resource Measurement Facility) and to all subsequent releases until otherwise indicated in new editions or technical newsletters. The previous edition still applies to RMF Version 3 Release 2 and may now be ordered using the temporary order numbers, LT68-1170 and LT68-1171. Changes are continually made to the information; 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 readers' 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 921-2, 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

This manual describes the program logic of the MVS/Extended Architecture (MVS/XA) Resource Measurement Facility (RMF). It is intended for persons who are debugging or modifying RMF. Because RMF is designed to execute on an MVS system, you should be familiar with the MVS system, especially the Real Storage Management (RSM), Auxiliary Storage Management (ASM), Input/Output Supervisor (IOS), and System Resource Management (SRM) components. These components of MVS directly support the operation of RMF.

The Resource Measurement Facility (RMF) is a measurement collection tool that is designed to measure selected areas of system activity and present the data collected in the form of SMF (System Management Facility) records, formatted printed reports, or formatted display reports. An installation can use the information in the RMF output to evaluate system performance and identify reasons for performance problems.

RMF gathers and reports data using three monitors: Monitor I, Monitor II, and Monitor III. An RMF monitor is a task that the user can invoke to collect and report on specific aspects of system performance.

- Monitor I runs in the background and measures data over a long period of time. It accumulates and stores the resource utilization data in SMF records.
- Monitor II runs in the background or at a display station, and provides a "snapshot" report of how resource use changes over a short period of time. Like Monitor I, it accumulates and stores the resource utilization data in SMF records.
- Monitor III measures contention of system resources, and the delay of jobs that such contention causes. It collects and reports data at a display station, and provides optional hardcopy reports.

How This Publication is Organized

The *RMF PLM* is divided as follows:

- *Volume 1*, which consists of Part 1 (LY28-1170) and Part 2 (LY28-1171), describes RMF control, Monitor I, Monitor II, and the RMF post processor.
- *Volume 2* (LY28-1172) describes RMF Monitor III.

This publication, Volume 1 contains the following sections:

- **Introduction** - an overview of each of the functions this publication documents.
- **Method of Operation** - a functional approach to each of the subcomponents. Each subcomponent begins with an overview diagram; all the individual diagrams applying to that subcomponent follow.
- **Program Organization** - a description of module-to-module flow for each subcomponent; a description of each module's function, including entry and exit. The module-to-module flow is ordered by subcomponent. The module descriptions are in alphabetic order without regard to subcomponent.
- **Data Areas** - control block overview diagrams; cross-reference to the microfiche.
- **Diagnostic Aids** - the messages issued, including the modules that issue, detect, and contain the message; register usage; return codes; abend codes; lock usage; error handling logic; and macro usage.
- **Appendix A** - an abbreviation list of acronyms used in the publication.

The following RMF information is available on microfiche:

- *Data Areas*, LYB8-1140.
- *Macro Usage Table*, LYB8-1141.
- *Symbol Usage Table*, LYB8-1142.

Associated RMF publications are:

- *MVS/Extended Architecture Resource Measurement Facility: Version 3 General Information Manual*, GC28-1115.
- *MVS/Extended Architecture Resource Measurement Facility: Reference and User's Guide*, LC28-1138.

The following MVS/XA publication provides information on MVS/XA components:

- *MVS/Extended Architecture System Logic Library* (multiple volumes). Volume 1, LY28-1208, contains order numbers for all volumes.

Contents

Figures

Summary of Amendments

Summary of Amendments
for LY28-1170-1 and LY28-1171-1
for RMF Version 3, Release 3

Additions and changes have been made to the *RMF Program Logic Manual* to support RMF Version 3 Release 3. These additions and changes support the IBM 3090 processor.

Volume I includes the following changed modules:

ERBMFMFC	ERBMFEDV	ERBRSPAG	ERBEXCHA
IGX00007	ERBMFEOQ	ERBRDEV	ERBEXENQ
ERBMFDTA	ERBMFRGM	ERBRIOQ	ERBEXIOQ
ERBMFIPG	ERBMFRPR	ERBRMFPP	ERBEXVSR
ERBMFIDV	ERBMFRDR	ERBMFP79	ERBMFIDA
ERBMFIOQ	ERBMFRQR	ERBMFPDU	ERBMFIDX
ERBMFIQA	ERBPUTSM	ERBDUIOQ	
ERBMFEAR	ERBGASD0	ERBMFXCB	
IGX00022	ERBGARD0	ERBEXCPU	
ERBMFDPP	ERBGSPAG	ERBEXPSP	
ERBMFDDP	ERBGDEV	ERBEXDEV	
ERBMFDOQ	ERBGIOQ	ERBEXPAG	
ERBMFEPG	ERBRASD0	ERBEXWKL	

The following modules have been added to this volume:

ERBCNFGF	ERBGIGQ
ERBMFDGQ	ERBRIGQ
ERBMFEGQ	ERBPPCON
ERBMFRGR	ERBDUIGQ
ERBPARSE	ERBEXIGQ

Summary of Amendments
for LY28-1170-0 and LY28-1171-0
as Updated October 31, 1983
by Technical Newsletter LN28-0886

Additions and changes have been made to the *RMF Program Logic Manual* to support RMF Version 3 Release 2.1. These additions and changes support RMF Version 3 Release 2.1. These additions and changes support the 4381 processor series.

Volume I includes the following changed modules:

ERBMFMFC	IGX00022
ERBCNFGC	ERBMFDOQ
ERBMFIZZ	ERBMFEOQ
IGX00007	ERBMFEVS
ERBMFTMA	ERBMFRQR
ERBMFIOQ	ERBGIOQ
ERBMFIDA	ERBRIOQ

The following modules have been added to this volume:

ERBCNFGG	ERBMFIQA

Summary of Amendments
for LY28-1170-0 and LY28-1171-0
as Updated August 31, 1983
by Technical Newsletter LN28-0857

Additions and changes have been made to the *RMF Program Logic Manual* to describe the new monitor, Monitor III, and the new Monitor I measurement of virtual storage. These additions and changes support RMF Version 3 Release 2.

Monitor III is described in a separate volume, Volume 2 (LY28-1172). This volume, Volume 1, also includes changed modules that support Monitor III and virtual storage measurement. The changed modules are:

ERBMFMFC	ERBMFRGM
ERBSESSC	ERBRMFPP
ERBMFINP	ERBMFPIR
ERBMFQOP	ERBMFPDU
MFIMAINL (Subroutine IGX00007)	ERBMFPLC
ERBMFTMA	ERBMFPLR
ERBMFIEQ	ERBMFPER
ERBMFEEQ	ERBMFXCB

A new module, ERBMFQSV, has been added to this volume to provide message and OCB service for the options module ERBMFQOP and ERB3CQOP (a Monitor III module described in Volume 2).

The following modules have been added to this volume to describe virtual storage measurement:

ERBDUVSR	ERBMFIVS
ERBEXVSR	ERBMFPVS
ERBMFDVP	ERBMFRES
ERBMFEVS	ERBMFRVR

Chapter 1. Introduction

The Resource Measurement Facility (RMF) is a tool that enables the user to obtain measurements of system activity through RMF sessions that collect, record, and report information about the system. The remainder of this section introduces RMF sessions and how the user controls them, and outlines the internal functional organization of RMF. For a complete description of how the user can control RMF processing, see the *Resource Measurement Facility (RMF) Reference and User's Guide*, SC28-1138.

RMF Sessions and How They Are Controlled

An RMF session is defined as a unique execution of RMF. Each type of RMF session accommodates a different type of system activity measurement and a different means of communicating that measurement to the user. There are three major classifications of sessions: Monitor I, Monitor II, and Monitor III.

Monitor I Sessions

A Monitor I session, also called a ZZ session, measures the following classes of system activity:

1. Processor activity
2. Paging activity
3. Workload activity
4. Channel path activity
5. I/O device activity
6. ASM/RSM/SRM trace activity
7. Page/Swap data set activity
8. Enqueue activity
9. I/O queuing activity
10. Virtual storage activity

A Monitor I session generates output in the form of printed reports and/or SMF records. For a complete description of the SMF records see the *RMF Reference and User's Guide.*

There are two ways a Monitor I session can be started:

1. Start a Monitor I session at the time RMF is initialized.

2. After RMF has been initialized, issue a START-session command to start a Monitor I session.

Only one Monitor I session can be active at a time, but its processing can be changed during execution through the input field of the system MODIFY command.

There are three ways to end a Monitor I session:

1. Issue a system STOP command. This ends all active non-TSO RMF sessions.

2. Wait for the expiration of a time interval specified by the user when the session was started.

3. Issue a STOP-session command to stop the Monitor I session.

Monitor II Sessions

There are three types of Monitor II sessions:

1. Background session
2. Local 3270 display session
3. TSO display session

All three types of sessions measure the same areas of system activity; they differ in the output generated and in the way they are controlled. Monitor II sessions measure the following classes of system activity:

1. Address space activity
2. Paging activity
3. Real storage/processor/SRM activity
4. Channel path activity
5. I/O device activity
6. I/O queuing activity
7. Enqueue activity
8. Transaction activity
9. Domain activity
10. Page/Swap data set activity

Monitor II sessions can:

- Display reports for immediate inspection.
- Write SMF records for post-processing.
- Produce printed reports.

Background Session

A background session generates output in the form of printed reports and/or SMF records. A background session is started from the operator's console by a START-session command, and processing, such as the time interval between the reports, is controlled by options specified when the session is started. These options can be modified during the session. The session is stopped by an operator command or the expiration of a user-specified length of time for the session. RMF must be initialized before the session can be started.

Local 3270 Display Session

The local 3270 display session generates displayed reports and gives the user the option of getting hardcopy reports. A local 3270 display session is started from the operator's console by a START-session command. Once started, it is controlled by the terminal user through session commands. The session can be stopped by either the operator or the terminal user, but only the terminal user can modify its processing. RMF must be initialized before the session can be started.

TSO Display Session

A TSO display session also generates displayed reports and gives the user the option of getting hardcopy reports. A TSO display session is started when the TSO user issues the TSO command RMFMON. Once started, this type of session is controlled by the user through session commands. The session can be stopped or modified only by the terminal users. A TSO display session does not require that RMF be initialized.

Monitor III Sessions

There are two types of Monitor III reporter sessions:

1. Local 3270 display session
2. TSO display session

Both types of sessions gather and report data about contention for system resources. Each obtains data from a Monitor III gatherer session, which must be active. See Section 1 of Volume 2 for a complete introduction to RMF Monitor III.

The following figure illustrates the types and some characteristics of RMF sessions.

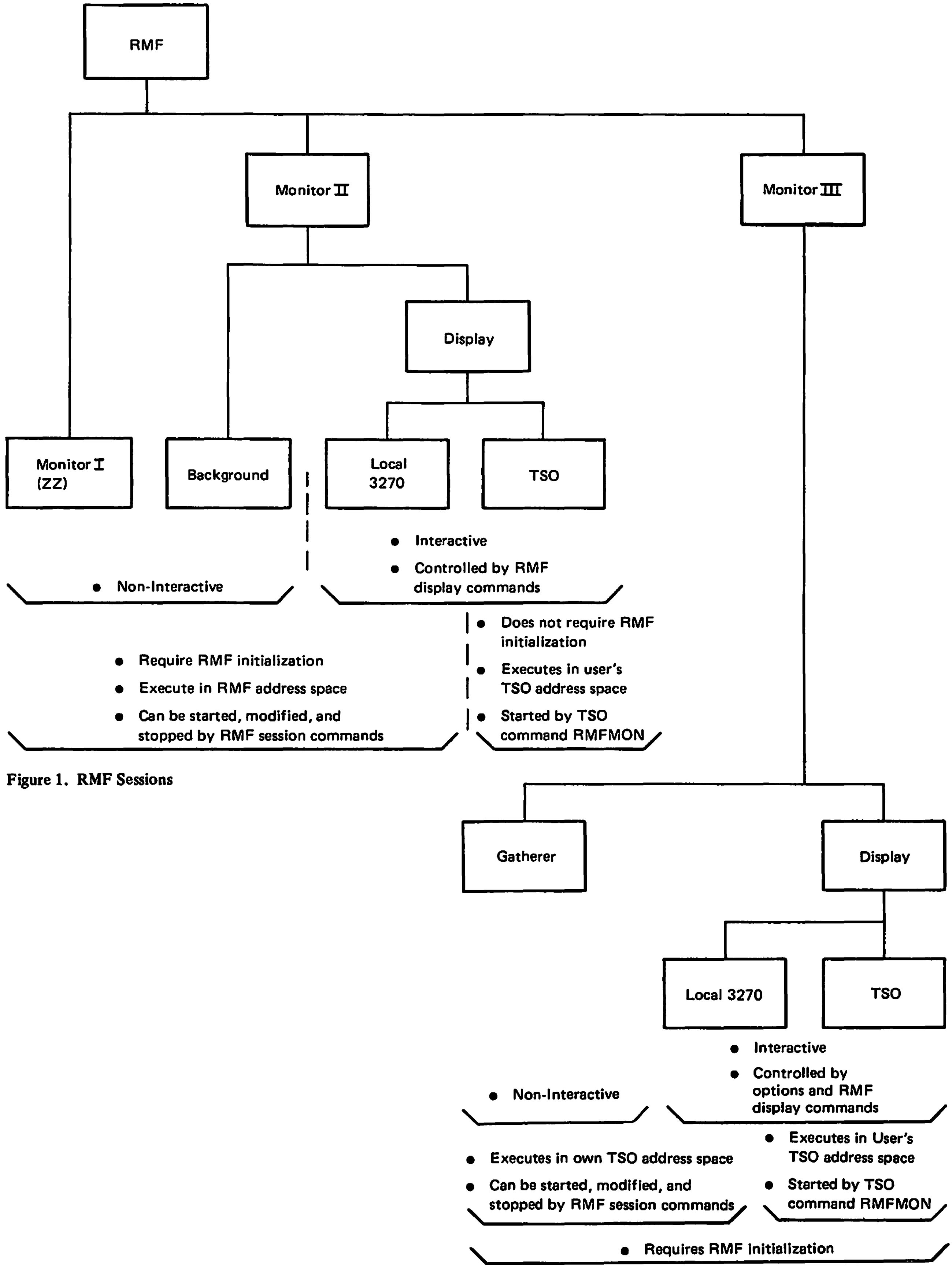


Figure 1. RMF Sessions

Internal Functional Organization of RMF

From an internal point of view, RMF processing is divided into five major functional sections: Measurement Facility Control; Monitor I Processing; Monitor II Processing; Post Processing; and Monitor III processing.

Measurement Facility Control (MFC)

RMF operates as a system task and executes under the control of those supervisory functions in MVS that dispatch and terminate work in the system. When dispatched, MFC Mainline (ERBMFMFC) receives control as the system task controlling RMF execution.

The MFC mainline calls the appropriate configuration table build routine (ERBCNFGC, ERBCNFGG, or ERBCNFGF, depending on the processor). The configuration table build routine reads the I/O configuration data from either the IOCDS (308X / IBM 3090) or the HSA (4381) and builds the internal tables that contain information about channels, devices, and logical control units. Channel initialization (ERBMFIHA), device initialization (ERBMFIDV), event arrival routine (ERBMFEAR), and I/O queuing modules (ERBMFIOQ, ERBMFEOQ or ERBMFEGQ, and ERBMFDOQ or ERBMFDGQ) use this information.

If a Monitor I session is being started at the time RMF is initialized, control passes from MFC Mainline to Session Create (ERBSESSC). In all other cases, control passes to Session Control (ERBMFCTL). The control block used for communication between MFC Mainline and Session Create/Session Control is the Application Control Table (ACT).

Session Create and Session Control call Input Merge (ERBMFINP) to merge the input from the various possible sources for Monitor I or Monitor II background sessions. (See the Extended Description of the ERBMFINP M.O. diagram for a complete description of the input sources.) Input Merge calls Queue Options (ERBMFQOP) and Lexical Analyzer (ERBLEXAN) to complete the parsing, then returns with the ACT pointing to a chain of Option Control Blocks (OCBs) built by Queue Options.

Session Create next passes control to the appropriate initialization module, ERBMFIZZ for a Monitor I session and ERBSESIT for a Monitor II session. Monitor I session initialization maps the contents of the OCBs into its own control blocks (MFSB, the session control block and MFPCT, the problem control table) and issues MFSTART SVC to start the appropriate measurements (see Monitor I Control).

For a display session, Monitor II Session Initialization builds a Picture Control Table (PCT) for each menu entry to be displayed to the user, then passes control to Display Process Control (ERBMFDPC). For a background session, Monitor II Session Initialization passes control to Background Process Control (ERBMFBPC) to control the processing of the reports indicated in the OCBs (see Monitor II Control).

Monitor I Processing

There are three parts to Monitor I processing: Monitor I Control, Monitor I Data Gathering (DG), and Monitor I Data Reporting (DR).

Monitor I Control

Monitor I Session Initialization (ERBMFIZZ) issues an SVC causing MFSTART Mainline (IGX00007) to get control. MFSTART Mainline passes control to Initialization Mainline (MFIMAINL) to get storage for and to initialize control blocks. (See the MFIMAINL M.O. diagram and the figure of the RMF Control Block Overview for the control block structure.) Initialization Mainline then branches to the initialization routines required by the input options. After initialization, MFIMAINL checks to see if any routines established ENF listening routines for changes in the state of a channel path, device, or channel measurement. If necessary, MFIMAINL issues the ENFREQP macro instruction to activate the event notification facility (ENF). MFSTART Mainline then causes control to be passed to Data Control (ERBMFDTA). Data Control is the entry into the Monitor I data gathering and data reporting modules. Upon return from Data Control, MFSTART Mainline branches to the Termination Processor (ERBMFTMA).

Monitor I Data Gathering (DG)

Monitor I DG collects information supplied by the various system components for eventual reporting through Monitor I Data Reporting; if required by the user, it also copies the information to the SMF data set. There are two types of DG routines - interval DG routines and cycle DG routines. There is an interval DG routine associated with each class of system activity; it is activated at reporting intervals to collect interval measurements and, optionally, to write an SMF record. Cycle DG routines are associated with each of the classes of system activity except workload, channel, and ENQUEUE. For any of these classes that are active, the cycle DG routines are entered a user-specified number of times within an interval. They sample queue lengths and maintain other intermediate data that the related interval DG routines summarize at reporting intervals.

Data Control issues an MFDATA SVC that causes the MFDATA SVC Mainline Processor (IGX00022) to get control. IGX00022 controls the operation of the interval DG routines. The MFROUTER SVC Processor (ERBMFEVT) controls the operation of the cycle DG routines. ERBMFEVT gets control when a user-specified cycle time elapses.

Monitor I Data Reporting (DR)

Monitor I DR produces all the formatted reports about the activities being monitored. There is one DR routine for each of the eight classes of system activity. The reports are written either at RMF termination or at the end of an interval, as requested by the user.

Data Control attaches Report Generator Control (ERBMFRGM) to:

- Allocate SYSOUT data space.
- Call the report generator for each report type requested.
- Free the interval measurement data space.

Monitor II Processing

A menu entry defines a measurement to Monitor II. In addition to supplying text for the display menu that is presented to the user, a menu entry contains functional data about the measurement: measurement name, a descriptive title, the names of the gatherer and reporter modules, and other data indicating the type and amount of data the measurement produces.

Monitor II Control

Monitor II Control begins with Monitor II Session Initialization (ERBSESIT). For a background session, Monitor II Session Initialization passes control to Background Process Control (ERBMFBPC), which calls Picture Build (ERBPCTBL) to build a Picture Control Table (PCT) for each report requested (indicated in the OCBs). The PCT contains all the data relating to the report. Background Process Control then handles the background session measurement collection.

For a TSO session, the RMFMON Command Processor (ERBMFTSO) gets control through the standard TSO command processor interface. RMFMON Command Processor builds an ACT and passes control to Session Create to create an RMF TSO session. Session Create attaches Monitor II Session Initialization, and from then on the TSO display session and the local 3270 display session are initialized identically, as follows.

Monitor II Session Initialization passes control to Terminal Initialization (ERBTERMI) to create a screen work area (MFSW). The MFSW contains all the data required to communicate with the terminal (see the Extended Description of the ERBTERMI M.O. diagram for a complete description of the MFSW). Next, Monitor II Session Initialization calls Picture Build to build a PCT for each entry in the menu. Then Display Process Control (ERBMFDPC) gets control to communicate with the user and to control the data gathering for the reports requested. Display Process Control uses Putline (ERBRMFPL) and Terminal Write (ERBTERMW) to communicate with the screen. Both Display Process Control and Background Process Control use Putline (ERBRMFPL) to generate hardcopy reports and Dynamic Allocation (ERBMFALL) to allocate their output data sets.

Monitor II Data Gathering and Data Reporting

A measurement in the Monitor II environment is made by a data gatherer module and a data reporter module. This separation of function allows one set of modules to operate in several environments: 1) display monitor, producing data at a display terminal and possibly hardcopy; 2) background monitor, producing printable or machine-readable data; or, 3) post-processor, producing printed reports from the machine-readable data.

Monitor II control loads the gatherer and reporter names in the menu. Those modules can be in SYS1.LINKLIB, SYS1.LPALIB, a steplib, joblib or any library in the linklist.

The means of communicating data between a gatherer and its reporter is an SMF record. This record has a header area common to all Monitor II records; a variable area, whose format is determined by the data being gathered, follows the header. The gatherer collects the data (it runs in key 8, problem state) and puts it in the SMF record. The reporter takes data from the SMF record, formats it for output, and then passes it to the Putline routine.

The user-specified operands passed to both the gatherer and reporter are used by each to produce the requested subset of data. (Default operands from the menu are also passed.)

Two user words and a subpool number are passed to gatherers and reporters. They may store the address of storage that subsequent invocations will use; for example, the gatherer could use the first word and the reporter could use the second; or one word could be used in addition to the SMF record to communicate data between the gatherer and reporter. (Note, however, that this communication is not possible between a gatherer called by the background monitor to produce SMF records and the reporter called by the post processor to handle the records.) Any storage that is to remain between executions of gatherers and reporters should be obtained from the subpool indicated. This subpool will be freed at session termination.

For a row report, the reporter is called with entry code 1, meaning it should only put out headers. It is then called with entry code 2, meaning to put out one line of data. For later consecutive executions of the row report, only the entry code 2 call is made.

For a table report, the reporter is called with entry code 2. On this call both headers and the full table of data are produced via many Putline calls.

The reporter is passed the address of the Putline routine, which is used to put out headers and data lines. Putline accepts single lines up to 79 characters and an indicator whether the line is a header or data. Putline accepts only two header lines per reporter execution. The number of data lines per execution should not exceed the number of relocate blocks contained in the SMF record.

Putline gets the data to the display device or to hardcopy or both, depending on the environment.

Putline issues return code 0 if everything is normal; it issues 4 or 8 if an I/O error occurred on hardcopy. If a limit was exceeded, such as too many data or header lines or an invalid length, a user abend is issued.

Post Processing

The Post Processor executes as a batch job that uses RMF SMF records as input and produces printed reports as output. The Post Processor Controller (ERBRMFPP) calls the Input Controller (ERBMFINC) to parse the input options from the SYSIN data set and to build a chain of Option Control Blocks (OCBs). The input controller uses Queue Options (ERBMFQOP) and the Lexical Analyzer (ERBLEXAN) to perform the parsing.

The Post Processor Controller calls all the routines needed to initialize the requested reports. The Post Processor reads the SMF records, selecting those required by the input options. The post processor controller calls ERBPPCOM to convert Version 2 records to the Version 3 format and ERBPPCON to complete the conversion to the format of the current level. Each selected record is passed to one or more report generating routines.

Report Generating Routines

- The Monitor I Interval/Duration Interface (ERBMFPIR) controls the production of interval reports. ERBMFPIR creates the Monitor I control blocks needed by Report Generator Control (ERBMFRGM). ERBMFPIR then calls ERBMFRGM, which calls the appropriate report generator for the SMF record being processed.

- The Monitor II interval reports are controlled by ERBMFP79. ERBMFP79 creates the Monitor II control blocks needed by the Monitor II data reporters and calls the appropriate data reporter.

- The Duration Report Controller (ERBMFPDU) calls duration collectors to accumulate data from several intervals into one SMF record. There is a collector for each type of duration report:

  ERBDUCPU - processor activity
  ERBDUPAG - paging activity
  ERBDUWKL - workload activity
  ERBDUCHA - channel path activity
  ERBDUDEV - I/O device activity
  ERBDUPSP - page/swap data set activity
  ERBDUIOQ - I/O queuing activity(308X/4381 processors)
  ERBDUIGQ - I/O queuing activity (IBM 3090 processors)
  ERBDUVSR - virtual storage activity

  At the end of a duration interval, the Duration Report Controller calls ERBMFPIR to produce a report from each accumulated SMF record.

- The Summary Report Controller (ERBMFPSC) collects data from Monitor I SMF records and generates a summary report.

- The Plot Report Controller (ERBMFPLC) collects data from Monitor I SMF records and calls the plot writer (ERBMFPLR) to produce the plots.

- The Exception Controller (ERBMFXCB) calls the exception processor (ERBEXCKP) to process specific exceptions. ERBEXCKP uses exception data handlers to determine if the exception being processed is met. There are several exception data handlers.

  ERBEXCPU - CPU Exception Data Handler
  ERBEXDEV - Device Exception Data Handler
  ERBEXPAG - Paging Exception Data Handler
  ERBEXWKL - Workload Exception Data Handler
  ERBEXVSR - Virtual Storage Exception Data Handler
  ERBEXCHA - Channel Exception Data Handler
  ERBEXENQ - Enqueue Exception Data Handler

ERBEXPSP - Page/Swap Data Set Exception Data Handler
ERBEXIOQ - I/O Queuing Exception Data Handler for 308X/4381 Processors
ERBEXIGQ - I/O Queuing Exception Data Handler for IBM 3090 Processors

The Exception Interval Interface (ERBMFPER) controls the production of interval reports that are generated because of exception conditions. ERBMFPER creates the control blocks needed by Report Generator Control (ERBMFRGM). ERBMFPER then calls ERBMFRGM to produce the exception interval report.

After processing the exception, the Exception Controller calls ERBMFPER to generate the requested interval reports if the EXRPTS control statement was specified and the exception is met. The exception controller generates the exception report itself.

Monitor III Processing

See the Introduction to Volume 2 for a description of Monitor III processing.

Chapter 2. Method of Operation

This section uses diagrams (both HIPO format and logic format) and text to describe the functions performed by the Resource Measurement Facility (RMF). The diagrams emphasize functions performed rather than the program logic and organization. Logic and organization is described in "Section 3: Program Organization."

HIPO method-of-operation diagrams are arranged in an input-processing-output format: the left side of the diagram contains data that serves as input to the processing steps in the center of the diagram, and the right side contains the data that is output from the processing steps. Each processing step is numbered; the number corresponds to an amplified explanation of the step in the "Extended Description" box. The object module name and labels in the extended description point to the code that performs the function. The logic format for method-of-operation diagrams is explained in Figure 1.

Figure 2 explains the symbols used in the diagrams. Figure 4 through 13 are the overview diagrams.

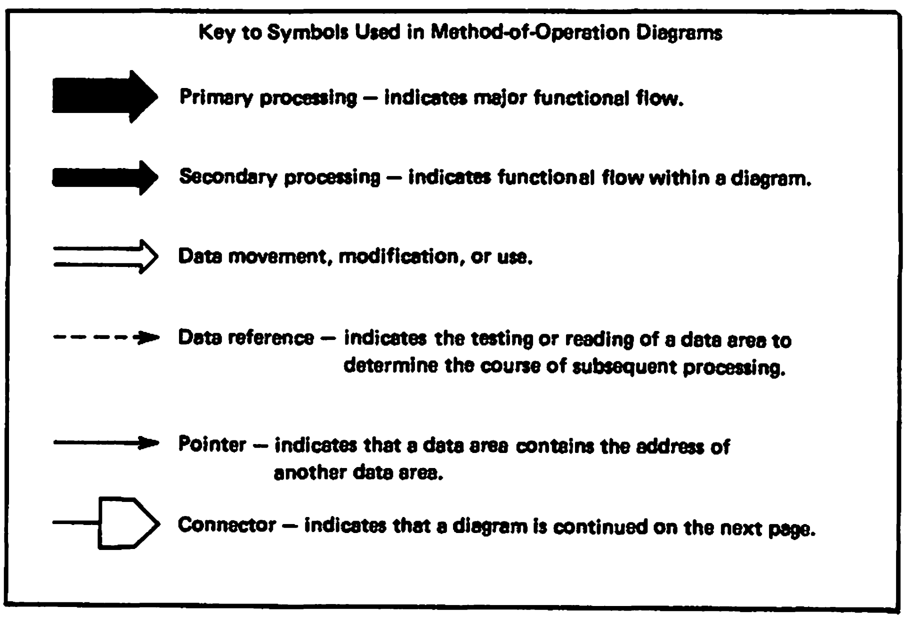


Figure 2. Method-of-Operation Diagram Symbols

The logic format detail information includes:

- Module description, which contains the following:
  - Descriptive name
  - Function of the module
  - Entry point names
  - External references, including routines, data areas, and control blocks
  - Tables

  For each entry point, the following information is listed:

  - Purpose of the entry point
  - Linkage
  - Callers
  - Input
  - Output
  - Exit normal
  - Exit error

- Module operation, which explains how the module performs its function.

- Diagnostic Aids, which provides debugging information. Major topics are:
  - Entry point names
  - Messages
  - Abend codes
  - Wait state codes
  - Return codes for each entry point. Within each entry point, the return codes might be further categorized by Exit Normal and Exit Error.
  - Entry register contents for each entry point
  - Exit register contents for each entry point

- Logic diagram, which shows the processing of the module, the input it uses, the output it produces, and the flow of control. Figure 3 illustrates the graphic symbols and format used in the logic diagrams. Circled numbers in the figure relate explanatory notes to the section of the illustration.

LOGICKEY - Key to the Logic Diagrams STEP 01

Callers

LOGICKEY

This paragraph describes what this module does. The same text appears under the FUNCTION heading on the Module Description page.

01 Numbered steps describe the processing at a high level.

A. Lettered steps describe the processing at a lower level.

SPQA

SPQAADQE SPQAEDQE

SPQE

SPQENEXT SPQESPQA

TCB

TCBPKF

02 Input and output fields.

The control block acronym or data area name appears above the input and output boxes, and the field names appear within the boxes. A dotted arrow means the data is referenced; a solid arrow means the data is modified.

SPQE

SPQENEXT
SPQESPQA
SPQETCB
SPQESPID
SPQEKEY
SPQESHR
SPQEOWN

SPQA

SPQAFADQ
SPQALADQ
SPQAFEDQ
SPQALEDQ

03 Calls an external routine passing the parameter TROB.

ITRFBR
TROB

04 Calls an internal subroutine (at the step indicated) passing two parameters.

SUBROUTN: 12
EFMSG1, TFWAFMSG

EAECB

EAERIMWT

ASCB

CVT

CVTBRET

TOB

TOBAASCB

05 Issues a macro instruction with these keywords, parameters, and options.

POST
(EAERIMWT, RC0) ASCB(TOBAASCB->ASCB) ERRET(CVTBRET)

06 Branches to the internal label at the step indicated.

>BRLABEL: 08

Figure 3 (Part 1 of 2). Key to Logic Diagrams

 LY28-1170-1

LOGICKEY - Key to the Logic Diagrams   STEP 07

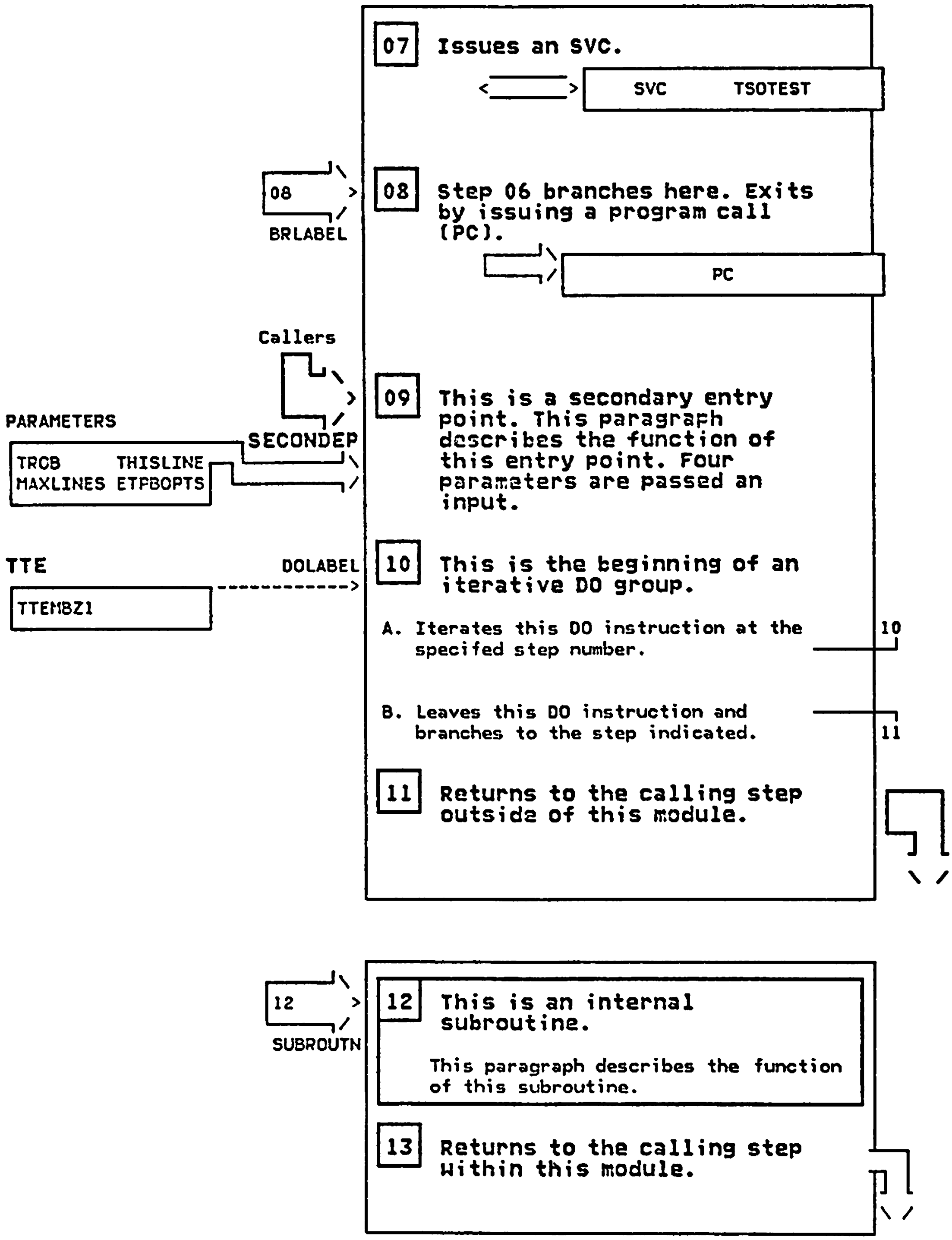


Figure 3 (Part 2 of 2). Key to Logic Diagrams

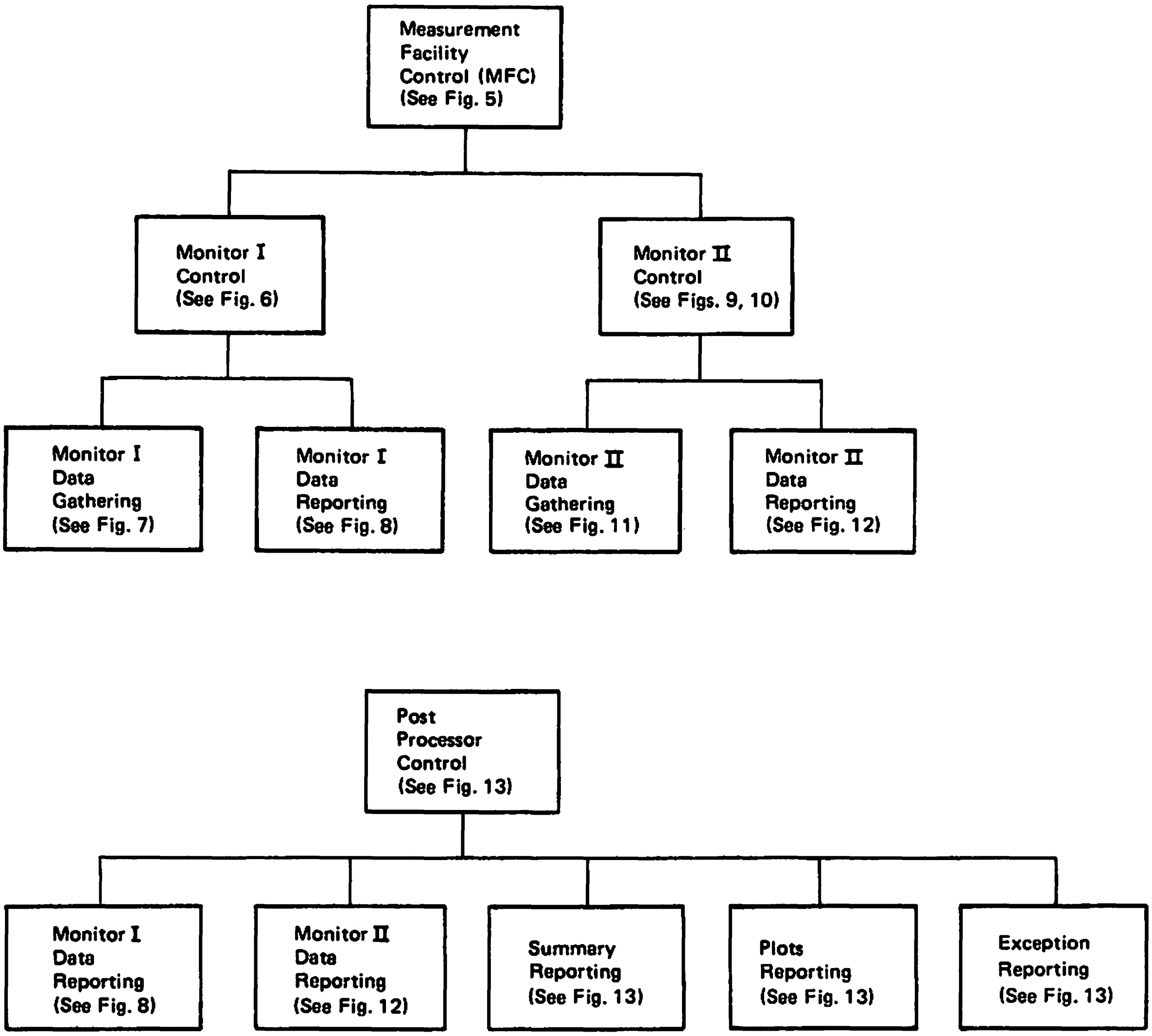


Figure 4. RMF Visual Table of Contents

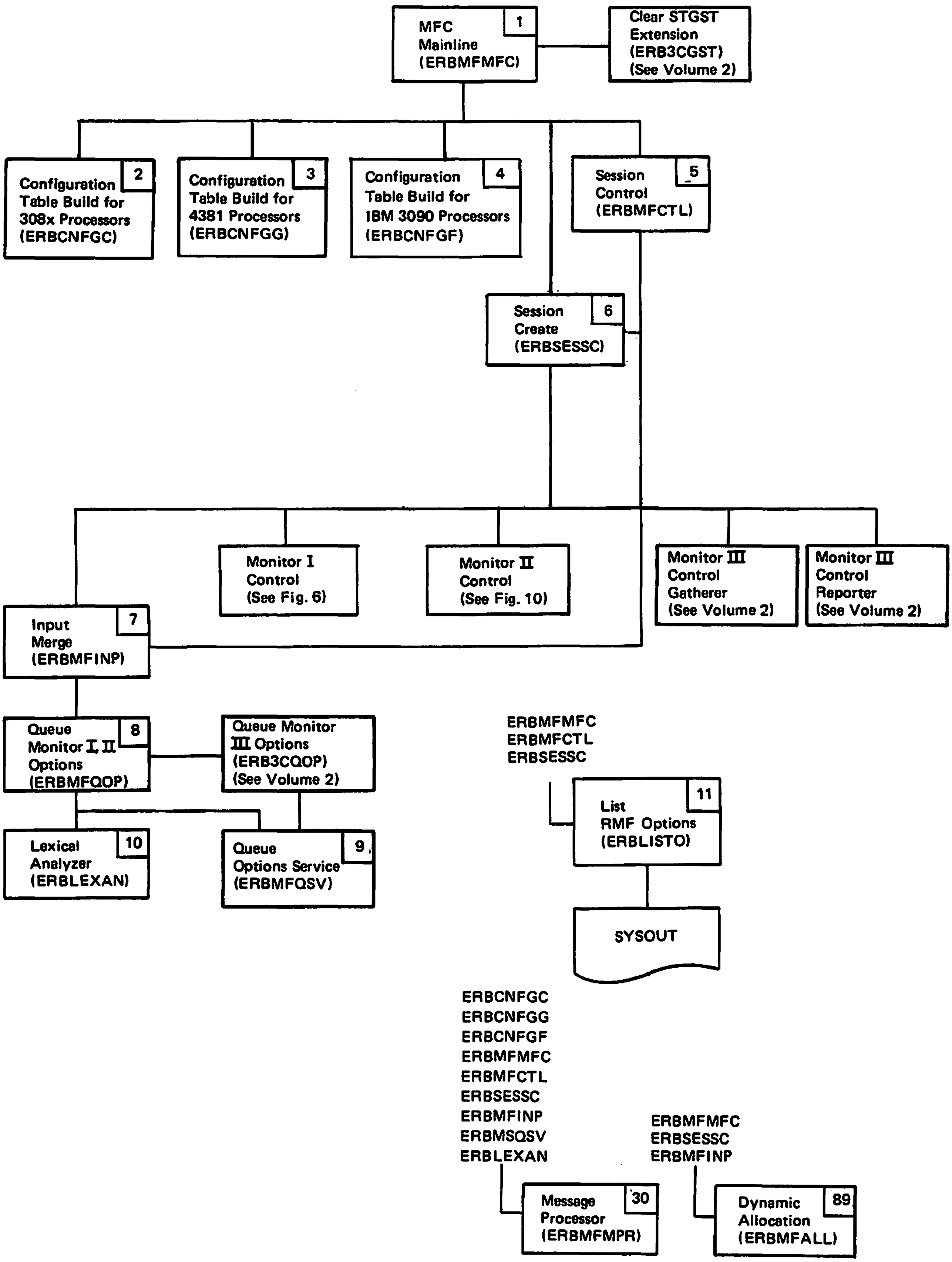


Figure 5. Measurement Facility Control Overview

Diagram 1. Measurement Facility Control (MFC) Mainline (ERBMFMFC) (Part 1 of 6)

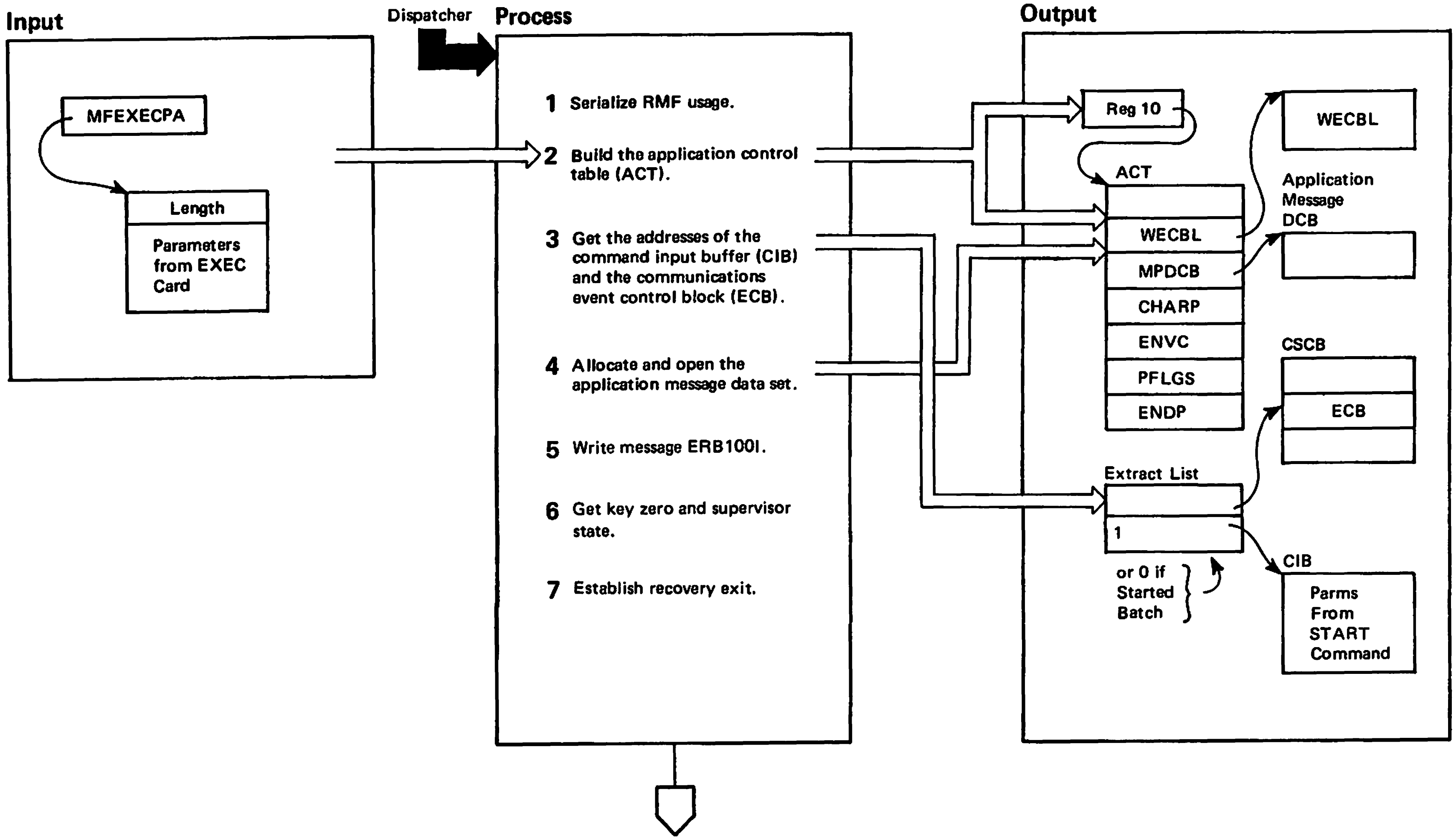


Diagram 1. Measurement Facility Control (MFC) Mainline (ERBMFMFC) (Part 2 of 6)

Extended Description	Module	Label
ERBMFMFC is the first RMF module to gain control upon starting RMF. It initializes the application, calls ERBSESSC to create a ZZ session if "NOZZ' was not coded on the start command, passes control to ERBMFCTL to control the application, and upon return from ERBMFCTL (termination of RMF) performs cleanup and exits.		
1 Enqueue on SYSZRBMF.ACTIVE to allow only one RMF task active at a time. If RMF is already active (the Enqueue return code ≠ 0), call ERBMFMPR to write message ERB200I.	ERBMFMPR	
2 Obtain storage via GETMAIN for the application control table (ACT) from subpool 9. Initialize the ACT.		
3 EXTRACT the CIB and ECB addresses from the communications area.		
4 Call ERBMFALL to allocate the dynamic message data set.	ERBMFALL	
5 Call ERBMFMPR to write 'RMF ACTIVE' message. This message indicates the beginning of a RMF session.	ERBMFMPR	
6 Enter supervisor state zero key in order to allow a portion of this module to operate in privileged mode.		
7 Issue the ESTAE macro instruction to establish error recovery exit linkage to ABNDEXIT.		

Diagram 1. Measurement Facility Control (MFC) Mainline (ERBMFMFC) (Part 3 of 6)

Input

CVT
- CVTMFCTL

CSD
- CSDIOML

STGST
- STGSIOML

STGST
- STGSIOML

STGST
- STGSIOML

Process

8 If CVTMFCTL is zero, build the global supervisor table (STGST) and global supervisor table extension (GSTC3).

9 Otherwise, if CVTMFCTL is not zero, call ERB3CGST to clean up GSTC3.

ERB3CGST

10 Copies the I/O measurement level from the CSD to the STGST.

A. If RMF is executing on a 308x processor, calls module ERBCNFGC to build the I/O configuration table.

ERBCNFGC

B. If RMF is executing on a 4381 processor, calls module ERBCNFGG to build the I/O configuration table.

ERBCNFGG

C. If RMF is executing on an IBM 3090 processor, calls module ERBCNFGF to build the I/O configuration table.

ERBCNFGF

Output

CVT
- CVTMFCTL

STGST
- STGGSTPT

GSTC3

STGST
- STGSIOML

Diagram 1. Measurement Facility Control (MFC) Mainline (ERBMFMFC) (Part 4 of 6)

Extended Description	Module	Label
8 Issue the GETMAIN macro to obtain storage for the global storage table (STGST) and the global storage table extension (GSTC3) from SQA subpool 245, unless these tables already exist. Initialize both tables and store the address of the STGST in the CVT.		
9 If both the STGST and the GSTC3 already exist, call ERB3CGST to clean up the global storage table extension.	ERB3CGST	
10 Copies the I/O measurement level from the CSD to the STGST. If RMF is executing on a 308x processor, calls module ERBCNFGC to build the I/O configuration table. If RMF is executing on a 4381 processor, calls module ERBCNFGG to build the I/O configuration table. If RMF is executing on an IBM 3090 processor, calls module ERBCNFGF to build the I/O configuration table.	ERBCNFGC ERBCNFGG ERBCNFGF	

Diagram 1. Measurement Facility Control (MFC) Mainline (ERBMFMFC) (Part 5 of 6)

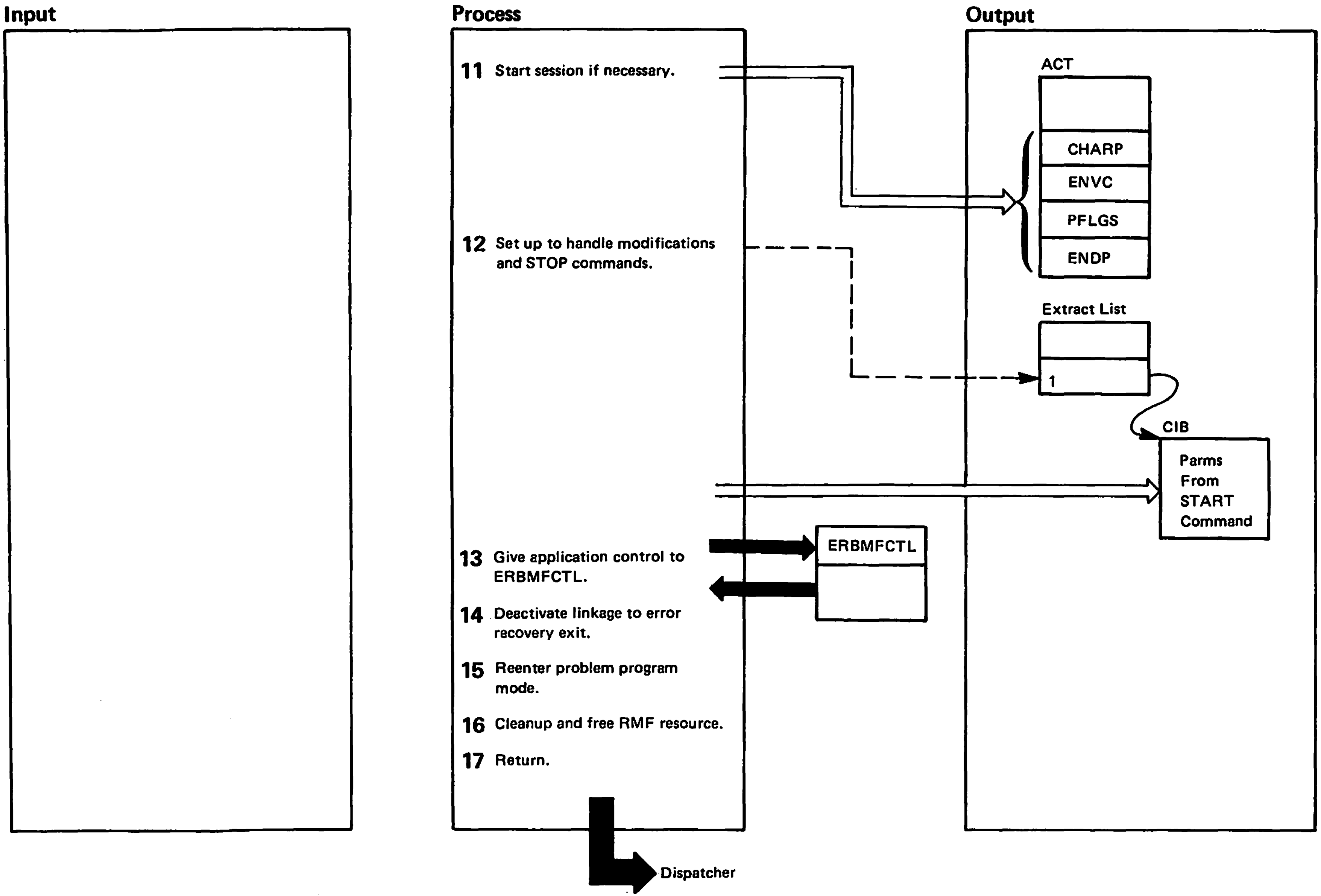


Diagram 1. Measurement Facility Control (MFC) Mainline (ERBMFMFC) (Part 6 of 6)

Extended Description	Module	Label
11 If this is a batch job, start the Monitor I (ZZ) session. If this job is being started from an operator's console and if NOZZ was not specified on the START command, call ERBSESSC to start the ZZ session.	ERBSESSC	
12 Delete the START CIB using QEDIT. Issue a second QEDIT to allow a maximum of five queued CIBs.		
13 Call ERBMFCTL. Control returns only when RMF is to terminate.	ERBMFCTL	
14 Issue an ESTAE macro instruction to deactivate the error recovery exit.		
15 Leave the supervisor state and reenter problem program mode.		
16 Release all storage that ERBMFMPR obtained from subpool 9. Call ERBMFMPR to write message ERB1021 (RMF terminated). Dequeue from RMF resources.		
17 Return to the dispatcher.		

Extended Description	Module	Label
Error Processing		
If an ABEND occurs, the exit routine ABNDEXIT gets control. ABNDEXIT places serviceability information in the System Diagnostic Work Area (SDWA) if there is one available, and in the VRA. An SDUMP is requested and global storage is freed.		
If the error occurred during Monitor I session initialization, ABNDEXIT requests a retry without starting the Monitor I session.		
At exit to RTM, ABNDEXIT requests recording of the error.		

Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 1 of 16)

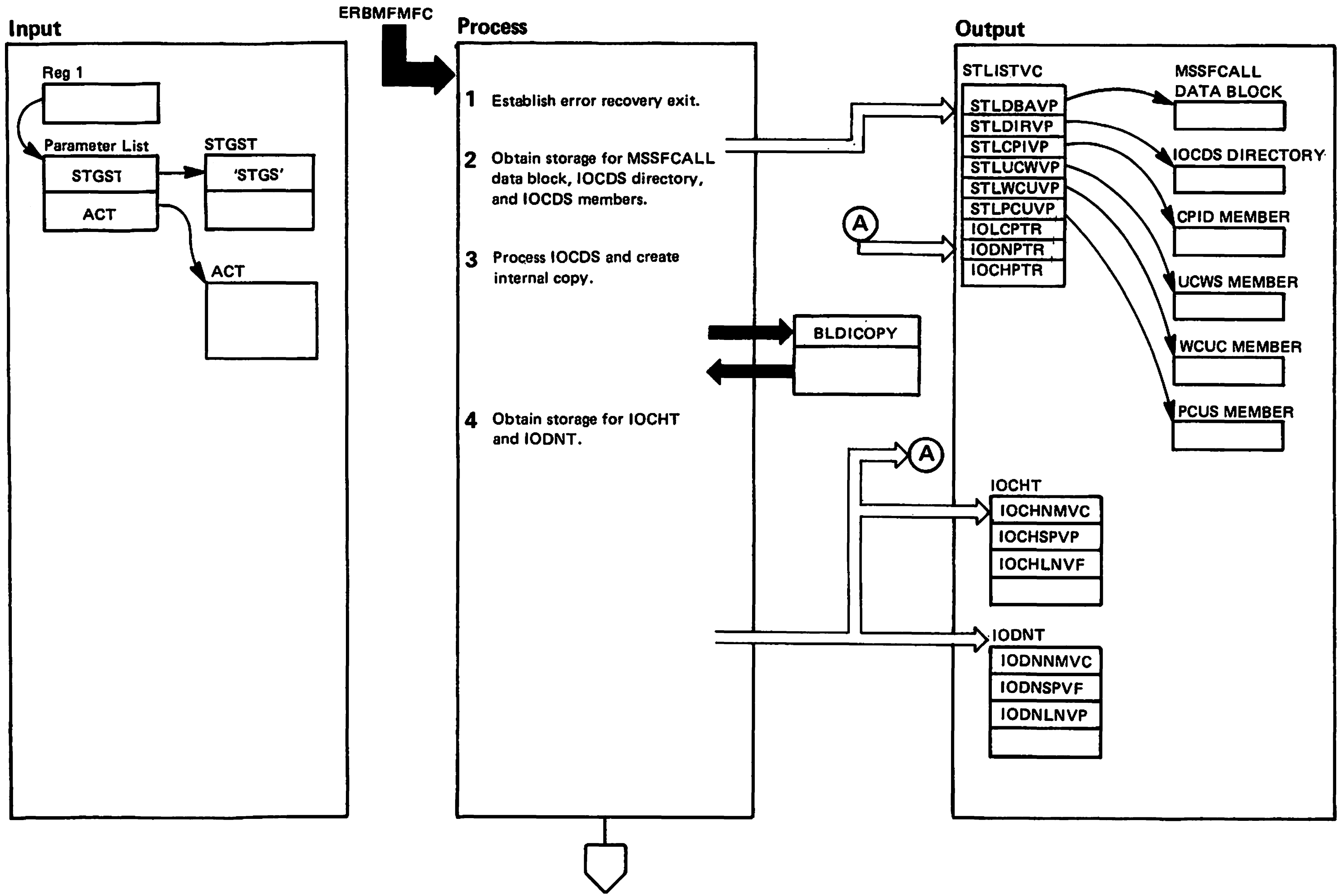
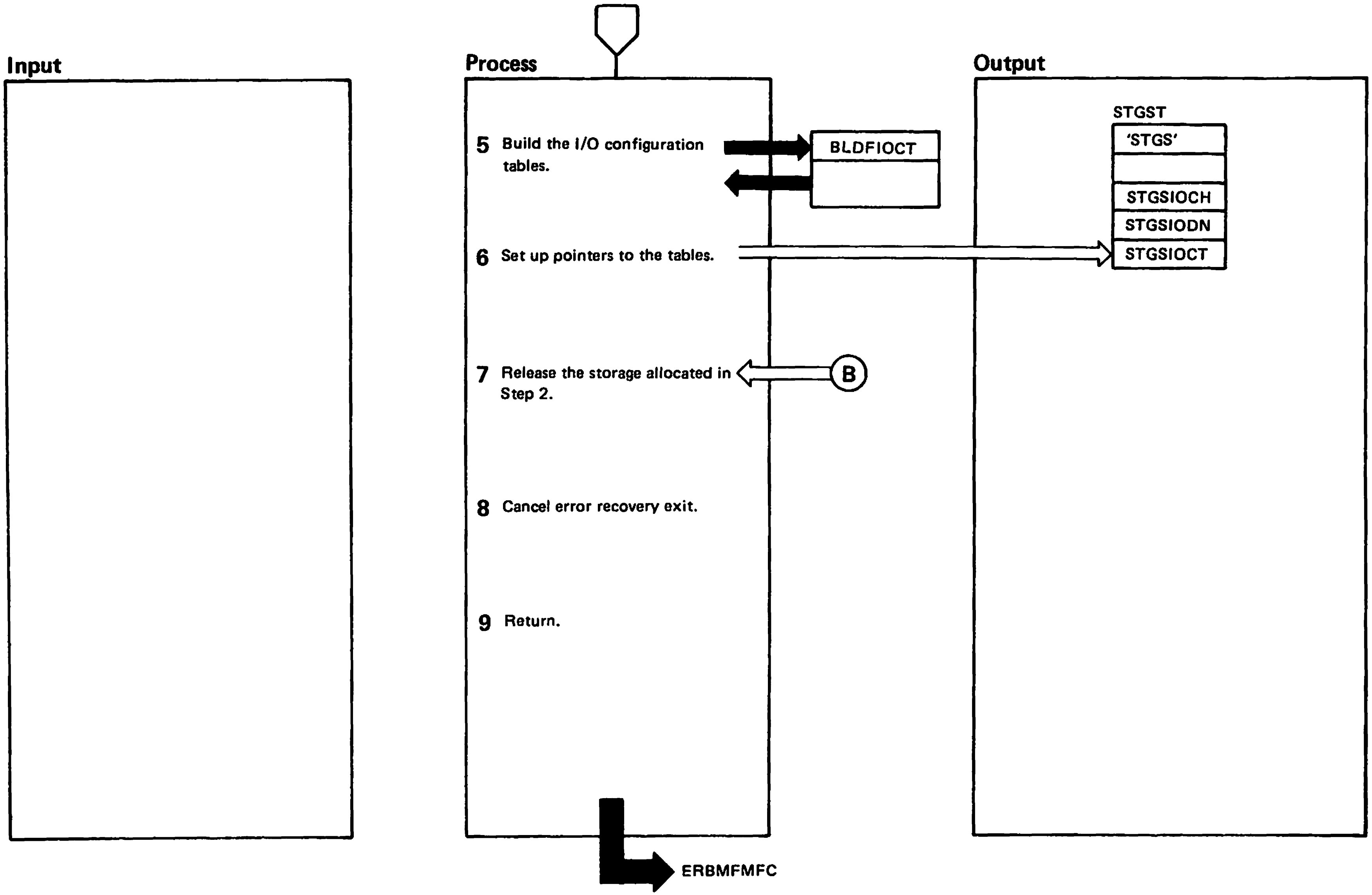


Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 2 of 16)

Extended Description	Module	Label
ERBCNFGC is called by ERBMFMFC to initialize the tables that are required by the I/O statistics gathering modules. It reads the input/output configuration data set (IOCDS) and creates the following tables: • Channel path table (IOCHT) • Device number table (IODNT) • Logical control unit table (LCUT)		
1 Issue the ESTAE macro instruction to establish linkage to an error recovery module (CNFGABND).		CNFGABND
2 Issue the GETMAIN macro instruction to obtain storage from subpool zero (SP0) for the following areas: • MSSFCALL data block • IOCDS directory • IOCDS CPID member • IOCDS UCWS member • IOCDS WCUC member • IOCDS PCUS member Save the addresses of these areas in the STLISTVC address list.		
3 Call the BLDICOPY subroutine to read the IOCDS and copy the data into the areas allocated in Step 2. If BLDICOPY returns a non-zero return code, issue one of the following error messages: • ERB265I for RC=4 • ERB266I for RC=8		BLDICOPY
Determine the condition of the IOCDS by testing the data set open bit. If the data set was not closed, call RDIOCDS to close it. Any error during the close operation causes error message ERB263I to be issued. Processing continues at Step 7.		RDIOCDS
4 Issue the GETMAIN macro instruction to obtain storage from subpool 245 for the channel path (IOCHT) and device number (IODNT) tables. Initialize the name, subpool number, and length fields of these tables.		

Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 3 of 16)



Extended Description	Module	Label
5 BLDFIOCT uses the internal copy of the IOCDS to build the I/O configuration tables:	BLDFIOCT	

- Channel path table (IOCHT)
- Device number table (IODNT)
- Logical control unit table (LCUT)

6 Store the addresses of the three configuration tables in the Global Supervisor Table (STGST).

7 Release the storage from subpool zero (SP0) that holds:

- MSSFCALL data block
- IOCDS directory
- IOCDS CPID member
- IOCDS UCWS member
- IOCDS WCUC member
- IOCDS PCUS member

8 Issue an ESTAE macro instruction to deactivate the linkage to the error recovery module (CNFGABND).

9 Return to ERBMFMFC with a return code in register 15.

Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 5 of 16)

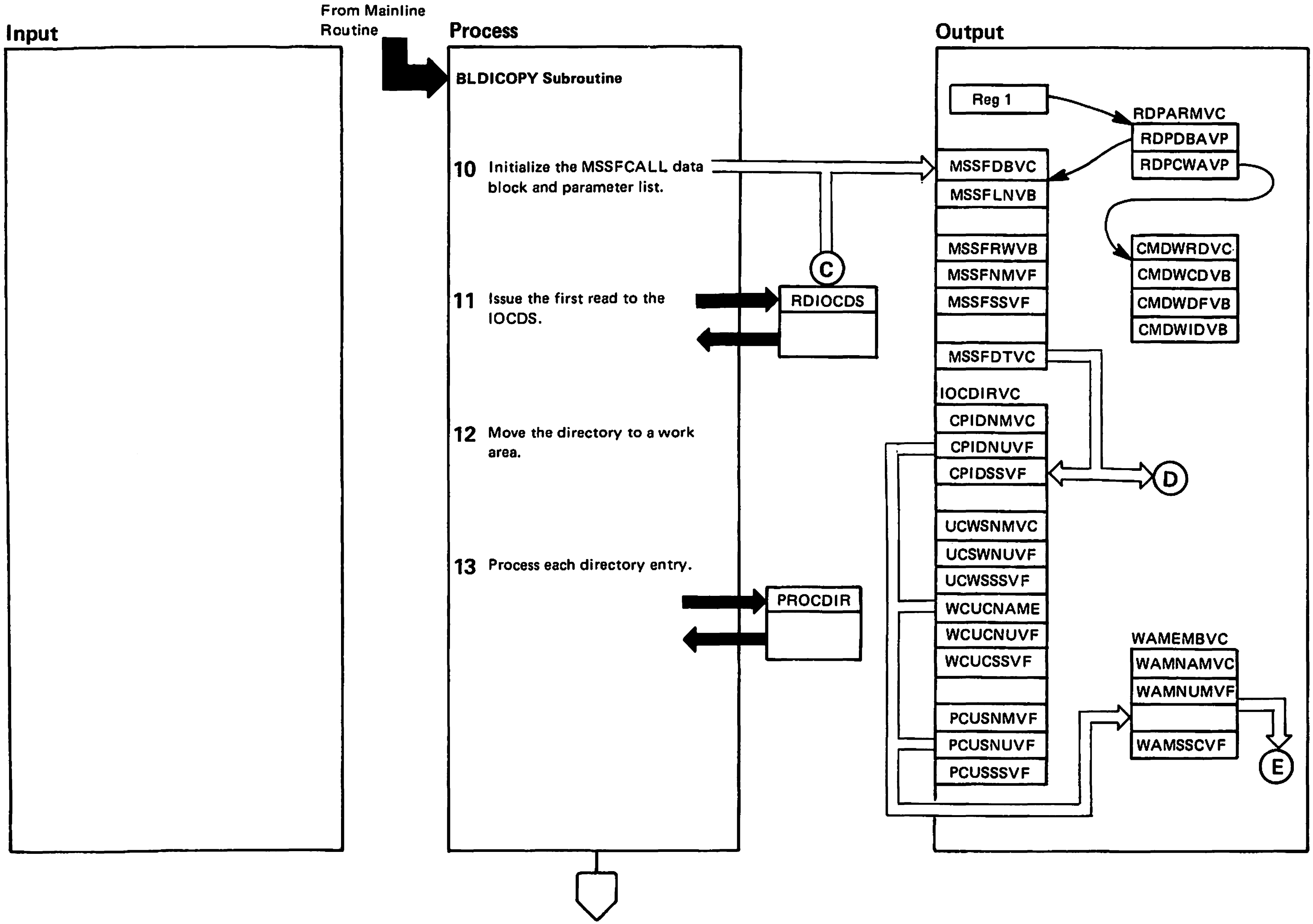


Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 6 of 16)

Extended Description	Module	Label
BLDICOPY Subroutine		
10 Load register 1 with the address of a two word parameter area containing the addresses of the MSSFCALL data block and the MSSFCALL command word. Initialize the MSSFCALL data block with the data block length, the read with open command word, a record count of 1, and a sector number of 0.		
11 Call the RDIOCDS to read the IOCDS. The first read opens the data set and transfers the contents of the IOCDS directory to the MSSFCALL data block.	RDIOCDS	
12 If the read was successful (RC=0), move the directory record to the storage obtained in Step 2, otherwise, processing continues at Step 17.		
13 Move the directory entry for the next IOCDS member to the work area, set an appropriate index and call the PROCDIR subroutine to add the entry to the control read table.	PROCDIR	
The above procedure is repeated until all the directory entries for the IOCDS are processed.		

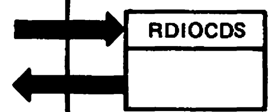
Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 7 of 16)

Input

Process

14 Prepare to read the next segment of the respective IOCDS member.

15 Issue a read to IOCDS.

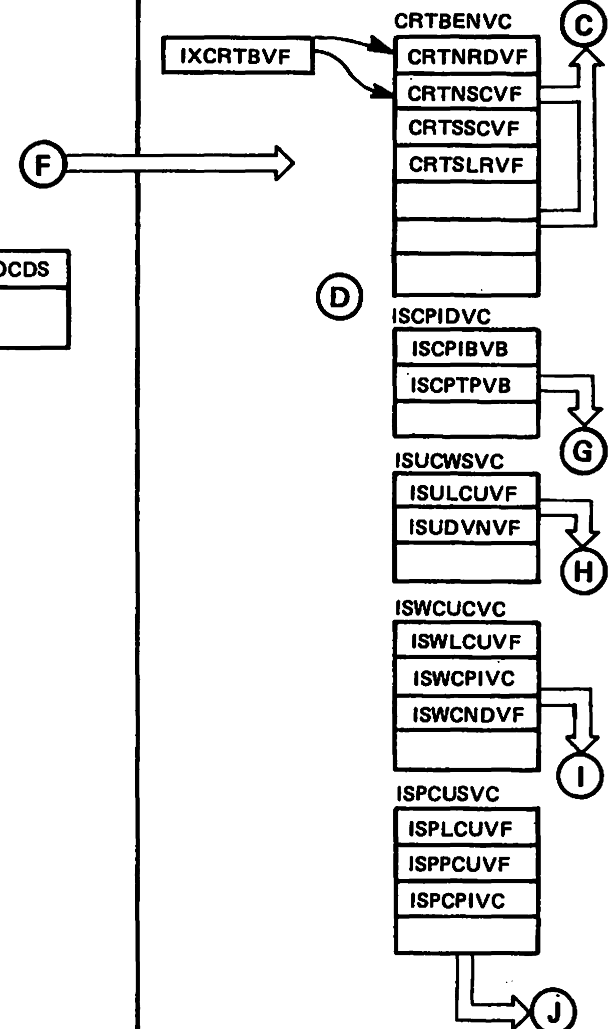


16 Process each IOCDS member.

17 Return.

Return to Mainline Routine

Output



Extended Description	Module	Label
14 Prepare to read the next segment from the respective IOCDS member using the information from the control read table built by PROCDIR. The values are inserted into the MSSFDBVC fields, used by the MSSFCALL interface. Set up to read seven sectors, except when the number of remaining sectors in an IOCDS member is less than seven. In this case, use the number of remaining sectors (CRTSLRVF).		
Set the addresses of MSSFDBVC and CMDWRDVC into the RDIOCDS parameter list RDPARMVC.		
15 Call the RDIOCDS subroutine to read sectors from the IOCDS. Set the command code to 'read normal' except for the last segment of the last IOCDS member. In this case, set the command code to 'read with close'.		
16 On return from RDIOCDS, if the read was successful, move the data contained in the MSSFCALL data block MSSFDBVC to the appropriate storage area (CPID, UCWS, WCUC, PCUS). The addresses for these areas are in the STLISTVC address list. Read all four members of the IOCDS by repeating Steps 14 through 16.		
17 Return to the mainline routine.		

Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 9 of 16)

Input

Process

From Step 11 or Step 15

RDIOCDS Subroutine

18 Initialize codes and wait count.

19 Read the IOCDS.

20 Set up appropriate return code and response code.

21 Return.

To Step 12 or Step 16

Output

RDPARMVC

RESPCDVF

RETCDEVF

Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 10 of 16)

Extended Description **Module** **Label**

RDIOCDS Subroutine

18 Initialize response code, return code, and wait count field.

19 Issue SVC 122 to read the IOCDS. Repeat read as long as the return code is 4 or 8 and the defined wait count is less than 60. Each time SVC 122 is issued, load the address of the parameter list (containing the address of the MSSFCALL data block and RMF command word) RDPARMVC into register 1. In addition, load register 15 with 6 (router code). Upon completion of the SVC, save the return code and, if the return code is 4 or 8, generate a wait of one second. Upon completion of the wait, add 1 to the wait count.

20 If the SVC 122 return code is 0 and the response code does not equal X'4020' set a return code of 4 in RETCDEVF. Save the MSSFCALL response code in RESPCDVF. In the SVC 122 return code equals 0 and the response code does equal X'4020' set a return code of 0. For all non-zero return codes from SVC 122, set a return code of 8. Save the actual SVC 122 return code in RCMSSFVF.

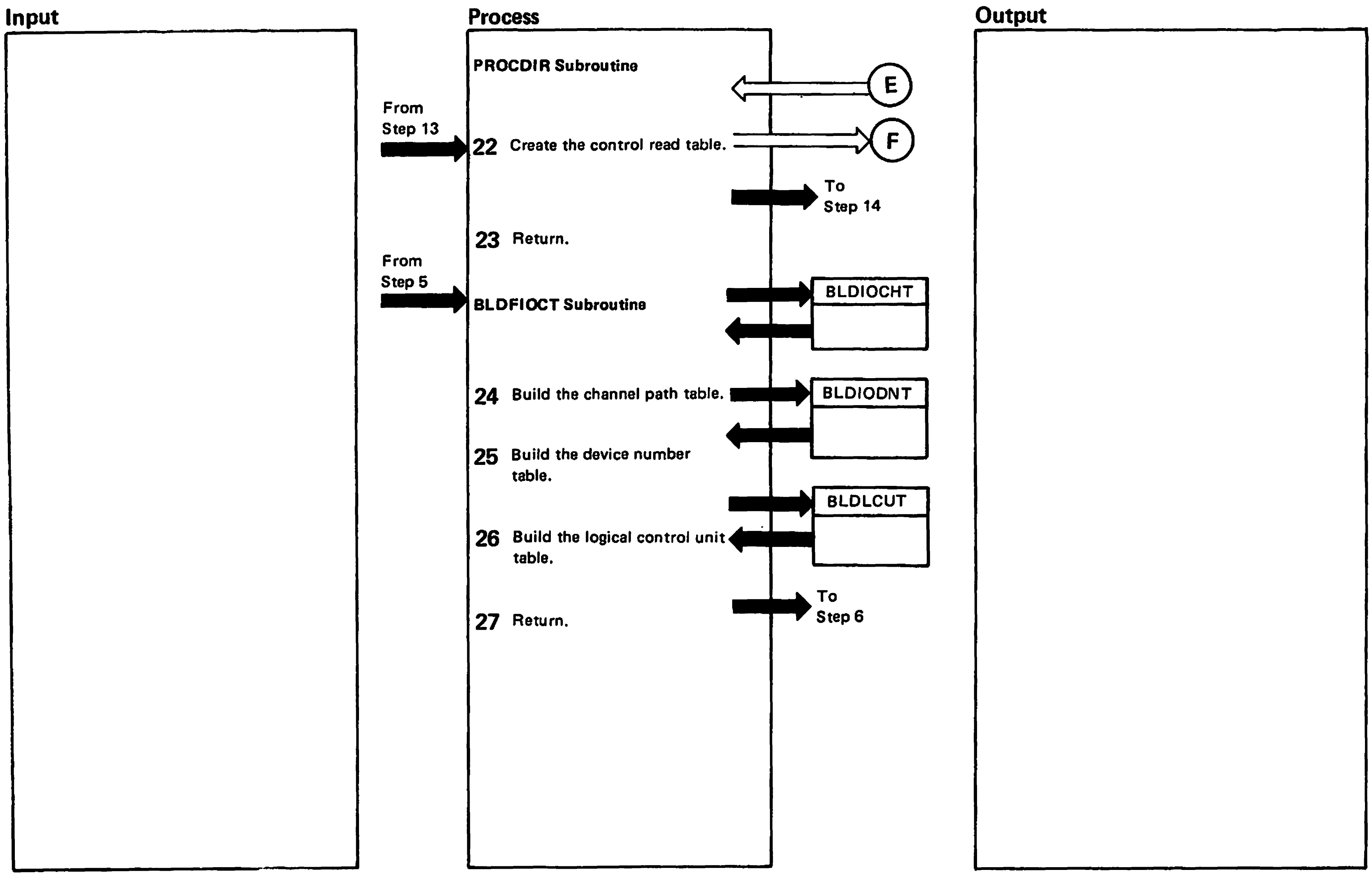
21 Return to the main routine, write one of the following codes:

0— IOCDS was read successfully, and the data is available in MSSFCALL 2K data block.

4— IOCDS could not be read, and the original MSSFCALL response code is saved.

8— IOCDS could not be read, and the original MSSFCALL return code is saved.

Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 11 of 16)



Extended Description	Module	Label

PROCDIR Subroutine

22 Store the number of records for each member (WAMNUMVF) and the starting sector number (WAMSSCVF) into the respective fields in the control read table (CRTBENVC). Calculate the number of reads per member by dividing the total number of sectors by 7. The maximum number of sectors than can be accessed by a read of 7. Use the remainder of the divide as the number of sectors to be accessed on the last read. If the remainder is non-zero, increment the number of reads by one. Otherwise, set the number of sectors to be accessed on the last read to 7.

Store the number of reads in CRTNRDVF and the number of sectors for the last read in CRTSLRVF.

23 Return to the mainline routine.

BLDFIOCT Subroutine

24 Call the BLDIOCHT subroutine to process IOCDS member CPID to create the channel path table.

25 Call the BLDIOCHT subroutine to process IOCDS member UCWS to create the device number table.

26 Call the BLDIOCHT subroutine to process IOCDS members WCUC and PCUS to create the logical control unit table.

27 Return to the mainline routine.

Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 13 of 16)

Input

From BLDFIOCT

Process

BLDIOCHT Subroutine

28 Initialize control pointers.

29 Build the channel path table.

G

30 Return.

To BLDFIOCT

From BLDFIOCT

BLDIODNT Subroutine

31 Initialize control pointers and IODNT.

32 Build the device number table.

H

33 Return.

To BLDFIOCT

Output

STLCPIVP

IOCHT
- IOCHNMVC
- IOCHSPVE
- IOCHLNVF
- IOCHIBVB
- IOCHTPVB

STLUCWVP

IODNT
- IODNNMVC
- IODNSPVF
- IODNLNVF
- IODNVBVB
- IODNLUVF

Extended Description	Module	Label

BLDIOCHT Subroutine

28 Set the start address for the CIPID member data to the first sector in the IOCDS storage area. Initialize the IOCHT index to zero.

29 Process all CPID member data by looping through all available sectors in the storage area. Each sector contains 64 entries, and each entry contains 4 bytes. For each valid non-zero entry, store the CPID installed bit and the channel path type into the IOCHT. For all invalid entries, leave the entry in the IOCHT as binary zeroes.

30 Return.

BLDIODNT Subroutine

31 Set the start address for the UCWS member data to the first sector of the IOCDS storage area. Initialize all the entries of the IODNT to binary zeroes.

32 Process all UCWS member data by looping through all available sectors in the storage area.

Each sector of the UCWS member consists of 16 entries, each 16 bytes, one for each defined device number. The entries are sequenced by UCW number from 0 to 4079. If the entry is not a dummy entry (not all bytes = 0), and the device number is valid (less than or equal X'FFF'), then use the device number as an index into the IODNT. Set the valid bit and store the associated logical control unit number into the entry. Calculate the total number of logical control units.

33 Return to the mainline routine.

Diagram 2. I/O Configuration Table Build for 308x Processors (ERBCNFGC) (Part 15 of 16)

Input

Process

BLDLCUT Subroutine

From BLDFIOCT

34 Allocate storage (SP9) for the logical control unit table.

35 Initialize the logical control unit table.

36 Process the IOCDS WCUC member and build part of the logical control unit table.

I

37 Process the IOCDS PCUS member and complete the logical control unit table.

J

38 Return.

To BLDFIOCT

Output

TNLCUVF

LCUT

LCUT
LCUTNMVC
LCUTSPVF
LCUTLNVF
LCUTTLVP
LCUTLCVF
LCUTNDVF
LCUTCPVF
LCUTCFVB
LCUTNPVF
LCUTPCVF

Extended Description	Module	Label

BLDLCUT Subroutine

34 Calculate the size of the logical control unit table using the following calculation:

(number of logical control units*length of LCUT entry)+length of LCUT header

Allocate storage for the logical control unit table from subpool 9 and store the address in IOLCPTR.

35 Initialize the logical control unit table to binary zeroes and set up the table header.

36 Process all WCUC member data by looping through all available sectors in the storage area and completing part of the LCUT.

Each sector of the WCUC member consists of eight 32 byte entries, one for each defined logical control unit. The entries are in ascending order by logical control unit number. Each entry contains up to four CPIDs and the number of devices associated with the logical control unit.

Each entry in the WCUC member is processed if it is not a dummy entry and the logical control unit number is less than or equal to the total number (=highest LCU) of logical control units in the system. The logical control unit number is used as an index to an entry in the LCUT. The logical control unit number and the number of devices associated with it are stored in the logical control unit entry. Then the CPIDs associated with the logical control unit are processed.

If the CPID is valid (not X'FF'), set the CPID valid flag and store the CPID through in the LCUT entry.

Extended Description	Module	Label

37 Process all PCUS member data by looping through all available sectors in the storage area and completing the LCUT.

Each sector of the PCUS member consists of 32 eight byte entries, one for each defined physical control unit. The entries are in ascending order by PCU number. Each entry is processed if it is not a dummy entry and the LCU number is less than or equal to the total number of LCUs (=highest LCU) in the system. The logical channel unit number is used as an index into the LCUT. All the CPID numbers are checked against the CPID numbers in the LCUT entry and, if a match occurs, then the physical control unit number is stored into the next empty entry for this CPID. The number of physical control units associated with this CPID is incremented by one.

38 Return to mainline routine.

Diagram 3. ERBCNFGG – Module Description

DESCRIPTIVE-NAME: I/O Configuration Table Build for 4381 Processors

FUNCTION:
The module builds I/O configuration tables for 4381 processors.
It builds the IOCHT, IODNT, and LCUT, using I/O configuration
data obtained from an MSSFCALL and STSCH interface.

ENTRY-POINT: ERBCNFGG

PURPOSE: See function

LINKAGE: BALR

CALLERS: ERBMFMFC

INPUT:
```
   Parameter list from caller:
   Parameter 1 - Pointer the the RMF Global Supervisor Table,
                 STGST.
   Parameter 2 - pointer the RMF Application Table ACT.
```

OUTPUT:
```
   If the return code is zero, the following tables are created:
   I/O device name table (IODNT)
   I/O channel path id table (IOCHT)
   Logical control unit table (LCUT)
```

EXIT-NORMAL: Returns to the caller via BR 14.

EXTERNAL-REFERENCES: See below

ROUTINES: None

CONTROL-BLOCKS:
```
   CVT      - Communication vector table
   IHAPSA   - Prefixed save area
   IHADCQ   - Device class queue
   IEFUCBOB - Unit control block (UCB)
   IECDIOCM - I/O communication block
   IECDIOSB - I/O supervisor block
   IHASCHIB - Subchannel information block
   IHASDWA  - System diagnostic work area
   IHAVRA   - Variable recording area
   ERBMFACT - Application control table
   ERBMFMID - Messages index table constants
   ERBSTGST - Global supervisor table
   ERBIOCHT - Channel path table
   ERBIODNT - Device number table
   ERBLCUT  - Logical control unit table
   ERBHSARB - Hardware system area request block
```

TABLES: XSTAB - internal sort table

ERBCNFGG - MODULE OPERATION

The ERBCNFGG module operates as follows:

1. Sets up addressability to the STGST and ACT, and sets up the recovery environment using the ESTAE macro.

2. Loops through the device class queue elements and accumulates the total number of devices in the system.

3. Calculates the size of, and obtains storage from subpool zero for, the internal sort table.

4. Loops through all UCBS, and, for each UCB, issues a read MSSFCALL (SVC 122) to retrieve I/O configuration data for the device. The I/O configuration data consists of physical control units (PCUs) and the channel path IDs (CHPIDs).

5. Sorts the internal sort table entries in ascending order of LCU number and device number.

6. Calculates the size of the IOCHT, IODNT, SCHIB, IOSB, and LCUT. Obtains storage for the IOCHT, IODNT, SCHIB, and IOSB from subpool 245, and obtains storage for the LCUT from subpool 9.

7. Loops through the internal sort table entries to build the LCUT. Performs the following processing when the LCU number changes:

   A. Updates the previous entry in the LCUT with the total number of devices for each LCU.

   B. Issues a store subchannel (STSCH) for the first device of this LCU. The CHPID and the path installed mask (PID) are the same for all devices for this LCU.

   C. For each CHPID, performs the following:

      1. Finds the CHPID in the SCHIB, and finds the corresponding PCU numbers from the I/O configuration data for this device.

      2. Indicates in the IOCHT table that this CHPID is installed, and saves the channel type in the IOCHT.

      3. Loops through the CHPID entries for the current LCUT entry to find this CHPID. Builds a new entry if the CHPID is not found.

      4. Loops through the PCU entries within the current CHPID entry to find the current PCU number. If the PCU entry is not found, uses the next free entry as the current entry, saves the PCU number, and updates the number of PCUs for this CHPID.

   D. Accumulates the total number of LCUs and the total number of devices per LCU.

   E. Turns on the valid indicator in the IODNT, and saves the LCU number in the IODNT.

9. Frees the storage used for the HSARB, the internal sort table, the SCHIB, and the IOSB.

10. Returns to caller.

ERBCNFGG - MODULE OPERATION (Continued)

RECOVERY-OPERATION:
Error recovery for this module is accomplished internally by ESTAE exit routine CNFGGESA at label CNFGGESA. The ESTAE exit routine issues an SDUMP, frees the SQA storage, and records the error.

ERBCNFGG - DIAGNOSTIC AIDS

ENTRY-POINT NAME: ERBCNFGG

MESSAGES:

ERB282I - RMF: IOCD INFORMATION UNAVAILABLE FOR xxxx
OF yyyy DEVICES. LAST RETURN/RESPONSE CODE ccc

RMF attempted to read the I/O configuration data for the device named in the message, but the MSSFCALL returned an unexpected error. The message includes either the response code or the return code from the MSSFCALL.

ABEND CODES: None

WAIT-STATE CODES: None

RETURN CODES:

EXIT NORMAL:

0 - The tables have been created, or if it was not possible to create the tables, all relevant table pointers are set to zero.

4 - If no devices were available to RMF, frees the storage for the internal sort table, zeros the pointers to the GSARB and the internal sort table, and returns to the caller.

REGISTER CONTENTS ON ENTRY:

Register 1 - Address of the parameter list
Register 13 Save area
Register 14 - Return address

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

Register 0-14 - unchanged
Register 15 - Return code

ERBMFMFC

ERBCNFGG

PARAMETERS

STGSTPTR ACTPTR

The module builds I/O configuration tables for 4381 processors. It builds the IOCHT, IODNT, and LCUT, using I/O configuration data obtained from an MSSFCALL and STSCH interface.

01 Sets up addressability to the STGST and ACT, and sets up the error recovery environment.

ESTAE
(GPR02P) PARAM=(GPR03P) MF=(E XESTAAVC)

DCQ

DCQFIRST DCQCHAIN
DCQUCBCT

02 Loops through the device class queue elements and accumulates the total number of devices.

03 Builds the internal sort table.

XSTAB

XSTAHDVC

ERBHSARB

HSARB

A. Calculates the size of the internal sort table.

B. Obtains storage for the sort table from subpool zero.

GETMAIN
(RU) LV(XTOTLNVF) BNDRY(PAGE) SP(ZEROCF) LOC(RES)

ERBHSARB

HSARB

C. Saves the pointers to the HSARB and the internal sort table, and clears the HSARB and the internal sort table.

D. Initializes the header for the internal sort table.

XSTAB

XSTAHMVC
XSTASUVF
XSTALEVF

E. Builds the internal sort table for all devices.

CNFGSRTB: 12

F. If no devices were available to RMF, frees the storage for the internal sort table, zeroes the pointers to the HSARB and the internal sort table, and returns to the caller with return code 4.

IOCHT

IODNT

04 Calculates the size of, and obtains storage from subpool 245 for, a storage area to contain the following:

- IOCHT
- IODNT
- SCHIB
- IOSB

GETMAIN
(RU) LV(IOTOTLVF) BNDRY(PAGE) SP(I245CF) LOC(RES)

IOCHT

SCHIB

IOSB

LCUT

05 Calculates the size of, and obtains storage from subpool 9 for, the LCUT.

GETMAIN
(RU) LV(LCUTLHVF) BNDRY(PAGE) SP(NINECF) LOC(RES)

ERBCNFGG - I/O Configuration Table Build for 4381 Processors STEP 06

Input:

IOCHT

IODNT

LCUT

SCHIB

XSTAB
XSTALEVF XSTAARVC

SCHIB
SCHPIM

ERBHSARB
HSARBY

SCHIB
SCHCHPID

XSTAB
XSTALEVF XSTAARVC

Process:

06 Initializes the headers for the IOCHT, IODNT, LCUT and IOSB.

07 Loops through the entries of the internal sort table to build the LCUT.

A. Sets the SCHIB to zero.

B. Issues a store subchannel (STSCH) for the first device of this LCU.
IOCSTSQE

C. Loops through the path installed mask (PIM) to find the CHID.

D. Updates the LCUT for this CHID
CNFGLCUT: 11

E. Updates the IOCHT for this CHID.

-Turns on the CHID installed indicator.
-If the channel type is HSARBY, turns on the byte multiplexor indicator.

F. Finds the internal sort table entry for the next LCU.

08 Loops through the internal sort table entries, and updates the IODNT for each entry.

09 Frees the storage for the HSARB, the internal sort table, the SCHIB, and the IOSB, and zeroes the pointers to these tables.

Output:

IOCHT
IOCHNMVC
IOCHSPVF
IOCHLNVF

IODNT
IODNNMVC
IODNSPVF
IODNLNVF

LCUT
LCUTNMVC
LCUTSPVF
LCUTLNVF

IOSB
IOSASID
IOSSYN
IOSSCHIB

IOSB
IOSUCB

IOCHT
IOCHIBVB
IOCHTPVB

LCUT
LCUTTLVF
LCUTNDVF

IODNT
IODNVBVB
IODNLUVF

SCHIB

IOSB

FREEMAIN

(RU) LV(XTOTLNVF) SP(ZEROCF) A((HSARBPTR))

FREEMAIN

(RU) LV(FRTOLNVF) SP(I245CF) A((SCHIBPTR))

10 Returns to the caller.

11

CNFGLCUT

11 Updates the LCUT for this CHID.

LCUT

LCUTCPVF LCUTC1VB

SCHIB

SCHCHPID

A. Sets up addressability for the CHID and PCU, and determines if the CHID is already available in the LCUT.

LCUT

LCUTLCVF

LCUT

LCUTPCVF

B. If the CHID is already available in the LCUT, determines if the PCU is already available within the CHID.

LCUT

LCUTPCVF

C. If the PCU is not available within the CHID, add a PCU.

LCUT

LCUTNPVF
LCUTPCVF

SCHIB

SCHCHPID

D. If the CHID is not available in the LCT, adds a CHPID and a PCU that refers to it.

LCUT

LCUTCPVF
LCUTC1VB
LCUTNPVF
LCUTPCVF

E. Returns to the subroutine caller.

ERBCNFGG - I/O Configuration Table Build for 4381 Processors **STEP 12**

12
CNFGSRTB

12 Builds the internal sort table entries for all devices.

ERBHSARB
HSARB
CVT
CVTUCBA
XSTAB
XSTAARVC

13 Initializes the MSSFCALL parameter list and the HSARB.

ERBHSARB
HSARLGTH

ERBHSARB
HSARRSPC HSARMDRC
HSARCD HSARFRMF
HSARRD

14 Loops through all UCBS, and performs the following processing for each UCB.

A. Issues the MSSFCALL (SVC 122) to obtain I/O configuration data. If the MSSFCALL sets a return code 4 or 8 (the MSSF is busy) re-issues the MSSFCALL every second for a maximum of 60 seconds.

SVC 122

ERBHSARB
HSARRSPC
HSARMDRC
HSARFCD
HSARSFCD
HSARSNUM
HSARCD

ERBHSARB
HSARRSPC HSARNC10
ERBHSARB
HSARCD HSARLCU
XSTAB
XSTANRVF
UCB
UCBCHAN

B. If the MSSFCALL completes successfully, updates the information in the internal sort table entry.

XSTAB
XSTANRVF

ERBHSARB
HSARRSPC
UCB
UCBNXUCB

C. If the MSSFCALL does not complete successfully, updates the count of the devices not available to RMF.

15 If the MSSFCALL failed once, issues message ERB232I.

A. Converts the number of devices unavailable to RMF, and the total number of devices, from binary to decimal.

ERBMFLMM
LMPART72

B. If the response code is zero, converts the return code from binary to decimal, and includes the return code in the message.

ERBMFLMM

LMPRT148

C. If the response code is not zero, converts the response code to external format, and includes the response code in the message.

ERBMFMPR

MID282I
VARTXVP(1)
ZEROCF
ACTMPDCB
'RMF'

XSTAB

XSTANRVF XSTAARVC

16 Sorts the internal sort table entries in ascending order of LCU number and device number.

17 Returns to the subroutine caller.

CNFGGESA

18 Provides error recovery for ERBCNFGG.

SDWA

SDWACWT

19 Determines if an SDWA is available. If an SDWA is not available, returns to RTM and does not attempt a retry.

SDWA

SDWAPARM

20 Restores the registers, and sets up addressability to the SDWA, HSARB, SCHIB, and IOSB.

SDWA

SDWAEAS

21 If an SDUMP was not previously requested, prepares to request an SDUMP.

ERBCNFGG - I/O Configuration Table Build for 4381 Processors **STEP 21A**

SDWA

SDWAMODN SDWACSCT SDWAREXN SDWAMLVL

IOCHT

IODNT

PARAMETERS

STGSTPTR

A. Saves the serviceability information in the SDWA and VRA.

B. Builds the list of storage areas for the SDUMP.

C. Requests an SDUMP.

SDUMP
HDRAD=DMPHDRCC LIST=DMPLSTVF SUBPLST=SPLSTCF SDATA=(LSQA SWA TRT PSA SUMDUMP) MF=(E SDMPAVC)

SDWA

SDWAMODN SDWACSCT SDWAREXN SDWACID SDWASC SDWAMLVL SDWARRL SDWACIDB

IOCHT

IOCHSPVF

IOCHT

IOCHLNVF

22 Frees the storage for the IOCHT.

FREEMAIN
(RU) LV(IOCHLNVF) SP(IOCHSPVF) A((IOCHPTR))

IODNT

IODNSPVF

IODNT

IODNLNVF

23 Frees the storage for the IODNT.

FREEMAIN
(RU) LV(IODNLNVF) SP(IODNSPVF) A((IODNPTR))

SCHIB

IOSB

24 Frees the storage for the SCHIB and IOSB.

FREEMAIN
(RU) LV(FRTOLNVF) SP(I245CF) A((SCHIBPTR))

25 Returns to the subroutine caller.

ERBCNFGF - MODULE DESCRIPTION

DESCRIPTIVE NAME: I/O Configuration Table Build for IBM 3090 Processors

FUNCTION:

ERBCNFGF builds I/O configuration tables for GP processors. It builds the channel path table (IOCHT), device number table (IODNT), and logical control unit table (LCUT) from data in the I/O configuration data set (IOCDS).

ENTRY POINT: ERBCNFGF

PURPOSE: See function

LINKAGE: BALR

CALLERS: ERBMFMFC during RMF initialization

INPUT:
The caller provides the following parameters:

STGSTPTR	Pointer to the STGST (RMF global supervisor table)
ACTPTR	Pointer to the ACT (RMF application control table)

OUTPUT:
The following tables are built:
IOCHT - Channel path table
IODNT - Device number table
LCUT - Logical control unit table

EXIT NORMAL: Return to caller

EXTERNAL REFERENCES:

ROUTINES: None

CONTROL BLOCKS:
ERBMFACT - Application control table
ERBMFMID - Messages index table constants
ERBSTGST - Global supervisor table
ERBIOCHT - Channel path table
ERBIODNT - Device number table
ERBLCUT - Logical control unit table
IHASCCB - Service call control block
IHASDWA - System diagnostic work area
IEEMSPCS - Service processor call SVC interface (SPCS)
IHAVRA - Variable recording area

ERBCNFGF - MODULE OPERATION

1. Establishes an ESTAE recovery routine (CNFGABND).
2. Obtains local storage from subpool 0 for reading and processing the IOCDS.
3. Calls subroutine BLDICOPY to process the IOCDS and create an internal copy of it.
4. If BLDICOPY read the IOCDS successfully, obtains global storage from subpool 245 for the IOCHT and IODNT.
5. If BLDICOPY read the IOCDS successfully, calls subroutine BLDFIOCT to process the IOCDS copy, to obtain storage from subpool 9 for the LCUT, and to build all I/O configuration tables.
6. If BLDICOPY did not read the IOCDS successfully, issues message ERB265I or ERB266I, depending on the codes returned. If necessary, closes the IOCDS. If the close fails, issues message ERB263I.
7. Frees the local storage obtained from subpool 0.
8. Deletes the recovery environment.
9. Returns to the caller with a return code of 0 in register 15.

RECOVERY OPERATION:

Error recovery for this module is accomplished internally by ESTAE exit routine CNFGABND at label CNFGABND. If an SDWA is provided, CNFGABND activates the DAE service, issues an SDUMP (if one was not requested previously), frees the global storage obtained from subpool 245 for the IOCHT and IODNT, and closes the IOCDS (if it is still open). Issues message ERB263I if the close fails and exits to RTM.

ERBCNFGF - DIAGNOSTIC AIDS

ENTRY POINT NAME: ERBCNFGF

MESSAGES:

ERB263I RMF UNABLE TO CLOSE IOCDS
RETURN|RESPONSE CODE ccc

ERBCNFGF attempted to close the IOCDS, but the close failed. The message includes the return or response code.

ERB265I IOCDS INFORMATION UNAVAILABLE TO RMF
RESPONSE|RETURN CODE cccc

ERBCNFGF encountered an error while trying to read the IOCDS. The message includes a response or a model dependent return code (MDRC).

ERB266I IOCDS INFORMATION UNAVAILABLE.
RETURN CODE ccc

ERBCNFGF encountered an error while trying to read the IOCDS. The message includes a return code.

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES: None

REGISTER CONTENTS ON ENTRY: Irrelevant

REGISTER CONTENTS ON EXIT: Irrelevant

ERBCNFGF - I/O Configuration Table Build for IBM 3090 Processors

STEP 01

ERBMFMFC during RMF initialization

PARAMETERS

STGSTPTR ACTPTR

ERBCNFGF

ERBCNFGF builds I/O configuration tables for IBM 3090 processors. It builds the channel path TABLE (IOCHT), device number table (IODNT), and logical control unit table (LCUT) from data in the I/O configuration data set (IOCDS).

01 Establishes an ESTAE recovery routine.

02 Obtains storage from subpool 0 for the SCCB 4K data block, the IOCDS directory, and the following IOCDS members: CPID, UCWS, LCUS, and PCUS. Saves the address of each area in the STLISTVC address list.

03 Calls subroutine BLDICOPY to read the IOCDS and create an internal copy of it.

BLDICOPY: 18

04 If no error occurred during the read of the IOCDS, obtains global storage from subpool 245 for the IOCHT.

ERBIOCHT

IOCHNMVC
IOCHSPVF
IOCHLNVF

05 If no error occurred during the read of the IOCDS, obtains global storage from subpool 245 for the IODNT.

ERBIODNT

IODNNMVC
IODNSPVF
IODNLNVF

06 If no error occurred during the read of the IOCDS, calls subroutine BLDFIOCT to use the IOCDS copy to create the I/O configuration tables.

BLDFIOCT: 38

07 If an error occurred during the read of the IOCDS, checks the error return code from BLDICOPY and issues the appropriate message.

IHASCCB

SCCBRSP

ERBMFLMM

LMFRT148

A. If a response code has been saved, issues message ERB265I, which includes the response code.

ERBMFMPR
MID265I, VARTXVP(1), NREPLYCF, ACTMFDCB, 'RMF'

ERBMFLMM

LMPART72

B. If the model-dependent return code (MDRC) indicates an error, issues message ERB265I, which includes the return code.

EREMFMPR
MID265I, VARTXVP(1), NREPLYCF, ACTMFDCB, 'RMF'

C. If the read of the IOCDS was unsuccessful, issues message ERB266I, which includes a return code.

ERBMFMPR
MID266I, CNFVTXVP, NREPLYCF, ACTMFDCB, 'RMF'

08 If an error has occurred, closes the IOCDS.

RDIOCDS: 26

IHASCCB

SCCBLNG

IHASCCB

SCCBRSP

ERBMFLMM

LMPART72 LMFRT148

09 If the close fails, issues message ERB263I with a return or response code.

EREMFMPR
MID263I, VARTXVP(1), NREPLYCF, ACTMFDCB, 'RMF'

10 Frees the storage obtained for the SCCB 4K data block, the IOCDS directory, and the IOCDS members (CPID, UCWS, LCUS, and PCUS).

11 Deletes the ESTAE recovery routine.

12 Returns to the caller, ERBMFMFC.

CNFGABND

13 Provides error recovery for ERBCNFGF.

14 If RTM does not provide an SDWA, then sets the no retry return code and returns to RTM.

SDWA

SDWAPARM

15 If there is an SDWA,

SDWA

SDWAEAS SDWAMODN
SDWACSCT SDWAREXN
SDWAMLVL

A. Places serviceability information in the SDWA.

B. Activates the DAE service.

C. Places serviceability information in the VRA. The addresses of the following tables are provided: STGST, IOCHT, IODNT, and LCUT.

ERBIOCHT

IOCHT

ERBIODNT

IODNT

PARAMETERS

STGSTPTR

D. Builds the address list for the SDUMP list parameter.

E. Initializes automatic storage and requests an SDUMP.

F. Issues the SETRP macro instruction to request recording, suppress the ABEND dump, and set the do-not-retry return code.

G. If the IOCHT and/or IODNT exist, frees SQA storage.

H. If the IOCDS is still open, closes it.

RDIOCDS: 26

SDWA

SDWAMODN
SDWACSCT
SDWAREXN
SDWACID
SDWASC
SDWAMLVL
SDWARRL
SDWACIDB

IHASCCB

SCCBLNG

IHASCCB

SCCBRSP

16 If the close fails, issues message ERB263I with a return or response code.

ERBMFMPR

MID263I, CNFVTXVP, NREPLYCF, ACTMPDCB, 'RMF'

SDWA

SDWARCDE

17 Returns to RTM.

18 BLDICOPY

18 Processes IOCDS data and creates an internal copy of each member needed by RMF.

19 Prepares to use the SCCB interface to read the IOCDS for the first time.

IHASCCB

SCCBLNG
SCCBFLAG

20 Issues a read with open to the IOCDS to obtain the directory.

RDIOCDS: 26

21 If the read was successful, uses the directory to create a control read table.

A. Processes the directory entry for the CPID member.

PROCDIR: 33

B. Processes the directory entry for the UCWS member.

PROCDIR: 33

C. Processes the directory entry for the LCUS member.

PROCDIR: 33

D. Processes the directory entry for the PCUS member.

PROCDIR: 33

22 If the first read was successful, prepares to read and process all IOCDS members that RMF needs.

23 Uses the SCCB interface to read the IOCDS.

RDIOCDS: 26

24 Stores the data from the following IOCDS members: CPID, UCWS, LCUS, PCUS. Closes the IOCDS.

RDIOCDS: 26

25 Returns to the subroutine caller.

IHASCCB

SCCBRSP

26 RDIOCDS

SPCS

SPCSESVC

26 Reads the IOCDS and verifies that the read completed successfully.

IHASCCB

SCCBRSP

27 Issues SVC 122 to read the IOCDS via the SCCB interface.

IHASCCB

SCCBRSP

28 Sets the return code to zero if the response code indicates that the read was complete and successful.

IHASCCB

SCCBRSP

29 If the model-dependent return code (MDRC) value is invalid for a good response code, sets the return code to 4. Saves the MDRC for error processing.

IHASCCB

SCCBRSP

30 If the MDRC was not invalid and a read reject response code was encountered, sets the return code to 4. Saves the response code for error processing.

31 If the read was not successful, sets the return code to 8. Saves the original return code from SVC 122 for error processing.

32 Returns to the subroutine caller.

33 PROCDIR

33 Processes the IOCDS directory and creates, for each member RMF needs, an entry in the control read table.

34 Stores the number of sectors and the starting sector number for each member in the control read table.

35 Calculates the number of IOCDS reads the member requires and stores the value in the control read table.

36 Calculates the number of sectors in the last read for a member and stores the value in the control read table.

37 Returns to the subroutine caller.

38 BLDFIOCT

38 Processes each member of the IOCDS read into RMF storage.

39 Calls subroutine BLDIOCHT to process the CPID member entries and build the IOCHT.

BLDIOCHT: 43

40 Calls subroutine BLDIODNT to process the UCWS member entries and build the IODNT.

BLDIODNT: 48

41 Call subroutine BLDLCT to process the LCUS and PCUS member entries and build the LCUT.

BLDLCUT: 52

42 Returns to the subroutine caller.

ERBIOCHT

IOCHNCCF

43 BLDIOCHT

43 Processes all entries in the IOCDS CPID member and builds corresponding entries in the channel path table (IOCHT).

ERBIOCHT

IOCHENVC

44 Loops through all sectors in the IOCDS CPID member and all entries in each sector.

45 Obtains the entry for each channel path identifier.

46 Processes the entry and stores the derived information (CPID installed bit and channel path type) into the corresponding IOCHT entry.

ERBIOCHT

IOCHIBVB
IOCHTPVB

47 Returns to the subroutine caller.

48 BLDIODNT

48 Processes all entries in the IOCDS UCWS member and builds corresponding entries in the device number table (IODNT).

ERBIODNT: IODNENVC

49 Loops through all entries in the IOCDS UCWS member.

50 If the entry is valid and the device number is not the highest device number (X'FFF'), then processes the entry. Stores the derived information (device is used/valid flag and the LCU associated with the device number) into the IODNT entry, using the device number as an index.

ERBIODNT: IODNVBVB, IODNLUVF

51 Returns to the subroutine caller.

ERBLCUT: LCUTHDVC LCUTENVC

52 BLDLCUT

52 Processes all entries in the IOCDS LCUS and PCUS members and builds corresponding entries in the logical control unit table (LCUT).

ERBLCUT: LCUTENVC

53 Obtains storage from subpool 9 for the LCUT.

ERBLCUT: LCUTNMVC, LCUTSPVF, LCUTLNVF, LCUTTLVF, LCUTENVC

54 Processes the LCUS member.

A. Loops through all sectors in the IOCDS LCUS member and all entries in each sector.

B. If an entry is valid, stores the derived information (CPID) into the LCUT, using the LCU number as an index.

ERBLCUT: LCUTLCVF, LCUTNDVF, LCUTCPVF, LCUTC1VB

55 Processes the PCUS member.

ERBCNFGF - I/O Configuration Table Build for IBM 3090 Processors

STEP 55A

ERBLCUT

LCUTCPVF LCUTC1VB
LCUTNPCF

ERBLCUT

LCUTNPVF

A. Loops through all sectors in the IOCDS PCUS member and all entries in each sector.

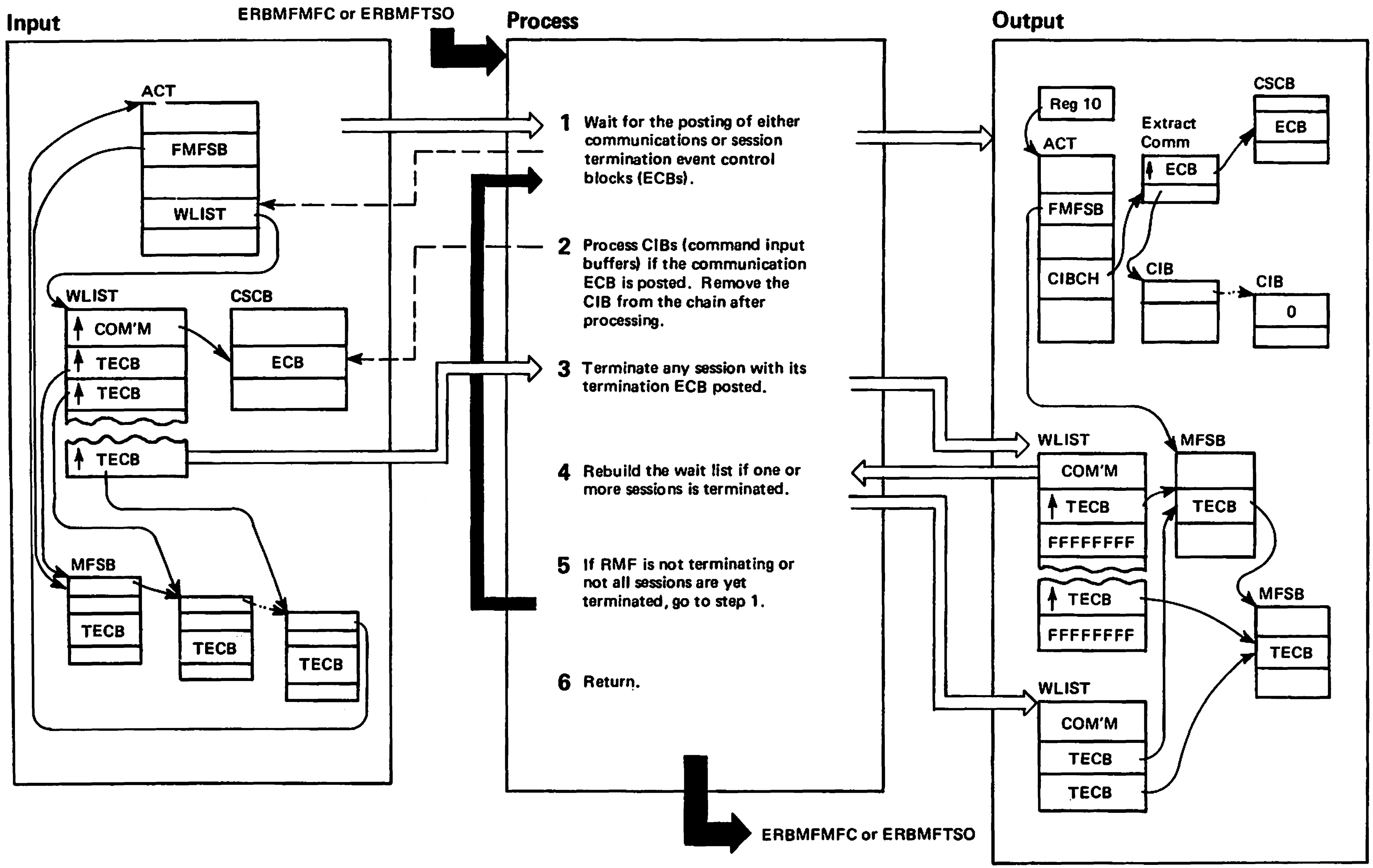
B. If an entry is valid, uses the LCU number as an index into the LCUT. If at least one CPID exists for an LCU, processes the entry. Stores the derived information (PCUS) into the LCUT.

56 Returns to the subroutine caller.

ERBLCUT

LCUTNPVF
LCUTPCVF

Diagram 5. Session Control (ERBMFCTL) (Part 1 of 2)



1 Wait for the posting of either communications or session termination event control blocks (ECBs).

2 Process CIBs (command input buffers) if the communication ECB is posted. Remove the CIB from the chain after processing.

3 Terminate any session with its termination ECB posted.

4 Rebuild the wait list if one or more sessions is terminated.

5 If RMF is not terminating or not all sessions are yet terminated, go to step 1.

6 Return.

Diagram 5. Session Control (ERBMFCTL) (Part 2 of 2)

Extended Description	Module	Label

ERBMFCTL controls the manipulation of sessions, that is, START, MODIFY, STOP of sessions and also the STOP of RMF.

1 ERBMFCTL issues WAIT pointing to Wait List.

2 The CIBDATA field of the CIB contains the session command and any parameters if a F RMF was issued. Subroutines within ERBMFCTL are invoked according to the command found:

F RMF,S .... ——→ CTLSSESS
F RMF,F .... ——→ CTLFSESS
F RMF,P .... ——→ CTLPSESS
F RMF,D .... ——→ CTLDSPLY
P RMF ——→ CTLPRMF

3 Look at each TECB pointed to by an entry on the WLIST. Call CTLTERMS to terminate the session of the MFSBTECB is posted.

4 Remove any ECB entries that were flagged by CTLTERMS for removal (all X'F's).

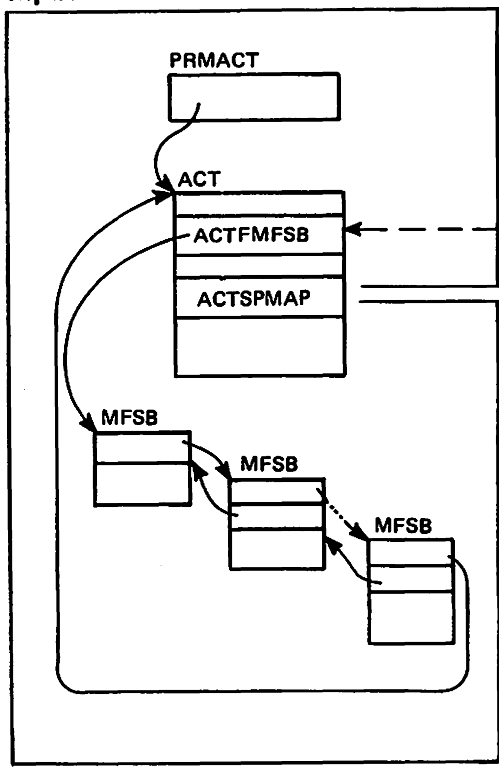
5 RMF continue until ACTTF = on (set by CTLPRMF or CTLTERMS) and all sessions have terminated.

6 Return code is always = 0.

Diagram 6. Session Create (ERBSESSC) (Part 1 of 4)

ERBMFCTL
or
ERBMFMFC

Input



Process

1 Determine whether this session is already active. If it is, go to step 7. Otherwise, continue with step 2.

step 7

2 Get storage for this session. If the maximum number of sessions (32) is already active, go to step 7. Otherwise, continue with step 3.

step 7

3 Get storage for the control blocks.

4 Initialize the session control block (MFSB).

5 If this session is a Monitor I session, a Monitor II gatherer session, or a reporter session, initialize the option control block (OCB) pool.

6 If this session is not a Monitor III reporter session:

a. Parse the options.

ERBMFINP

b. Allocate the session message data set.

ERBMFALL

c. Open the session message data set.

Output

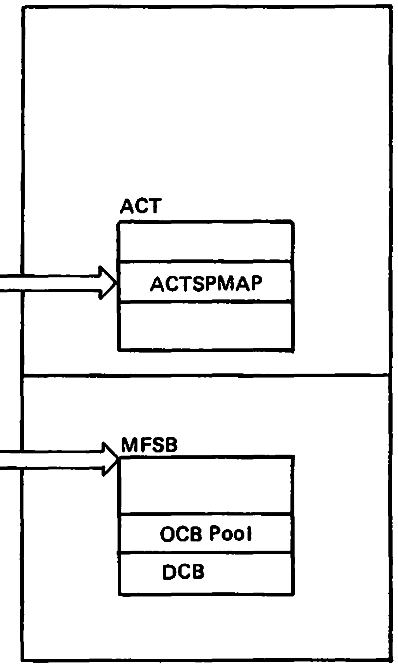


Diagram 6. Session Create (ERBSESSC) (Part 2 of 4)

Extended Description	Module	Label
ERBSESSC activates a session if:		
• the session name is unique, that is, a session with the same name is not already on the active chain, and		
• the maximum limit (32) of active sessions has not been reached.		
A session is activated as follows:		
a) Get a unique subpool for its control blocks.		
b) Get storage for the control blocks.		
c) Parse the command options (background sessions only).		
d) Attach the session initialization routine.		
e) Update the Wait ECB list.		
If a ZZ session is started automatically at the time RMF is started, control is passed to ERBSESSC from ERBMFMFC. All other entries are from ERBMFCTL.		
1 Search the session control block (MFSB) chain pointed to by ACTFMFSB. If this session name is contained in an MFSB, then this session is already active. Call ERBMFMPR to write message ERB200I.	ERBMFMPR	

Extended Description	Module	Label
2 Storage comes from subpool 7 for a ZZ session; storage comes from subpool 8 for a TSO session; and storage comes from subpool 6 for a Monitor III session. For Monitor II background and foreground sessions, the subpool is determined by an allocation bit within ACTSPMAP. In order to start one of these sessions, ACTSPMAP (a 32-bit field) is searched for a 0 bit. If a 0 bit is found, it is set to 1 and its location within ACTSPMAP is used to determine the subpool (10 + bit location in ACTSPMAP).		
3 The OCB pool and the message data set are required only for background sessions.		
4		
5		
6 a) Call ERBMFINP to parse the options.	ERBMFINP	
b) Call ERBMFALL to allocate the message data set.	ERBMFALL	
c)		
(Note: No option parsing and no message data set is provided for the Monitor III reporter.)		

Diagram 6. Session Create (ERBSESSC) (Part 3 of 4)

Process

7 Attach the appropriate session initialization routine:

ZZ session = ERBMFIZZ

Monitor III data gatherer session = ERB3CGAT

Monitor III data reporter local 3270 session = ERB3CREP

Monitor III data reporter TSO session = ERB3RCTL

Others = ERBSESIT

8 If initialization is successful, write the appropriate message and rebuild the wait list for ERBMFCTL.

Otherwise, write the appropriate message, and free the session subpools.

9 Return.

ERBMFCTL or ERBMFMFC

Output

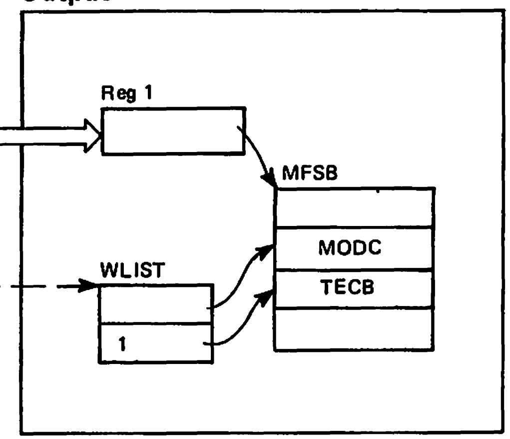
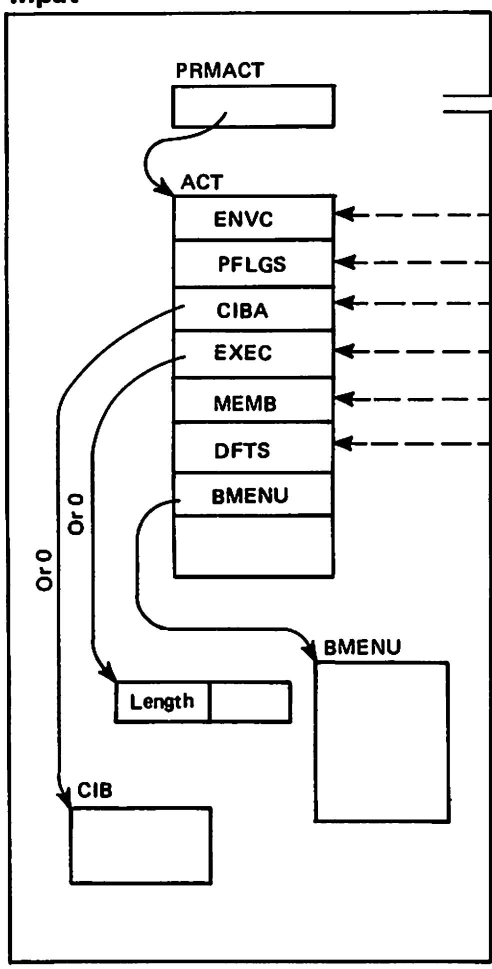


Diagram 6. Session Create (ERBSESSC) (Part 4 of 4)

Extended Description	Module	Label
7 Share the subpool assigned in step 2. MFSBTECB is posted automatically when ERBMFIZZ, ERB3CGAT, ERB3CREP, ERB3RCTL or ERBSESIT returns.		
8 Wait for either MFSBMODC or MFSBTECB to be posted. Posting of MFSBMODC is normal and indicates session initialization was successful. For this case, call ERBMFMPR to write message ERB100I (RMF active). If MFSBTECB is posted, initialization has failed and ERBMFMPR writes message ERB229I.	ERBMFMPR	
ERBMFMPR writes the following messages under the following conditions: • The session is already active (determined in step 1) – ERB200I. • The maximum number of sessions (32) is already active (determined in step 2) – ERB227I. • OPEN failed on the session message data set (step 6, c) – ERB226I.		
9 Return to caller with a return code = 0.		

Diagram 7. Input Merge (ERBMFINP) (Part 1 of 4)

Input



ERB3ROPT, ERBSESSC or ERBMFCTL

Process

1 Set pointers to the option table and, for the START command, to the defaults table.

2 Process the Command Input Buffer (CIB) parameters, if any.

3 Process EXEC card parameters, if any.

4 Process the default members, if necessary.

5 Process defaults, if necessary.

6 Validity check the options.

7 If operator intervention is allowed:

a. List the options on the terminal.

b. Process the operator reply.

c. Repeat steps a. and b. until the operator replies 'GO' and the validity check routine determines that no conflicts exist.

8 Return (return code = 0).

Caller

Output

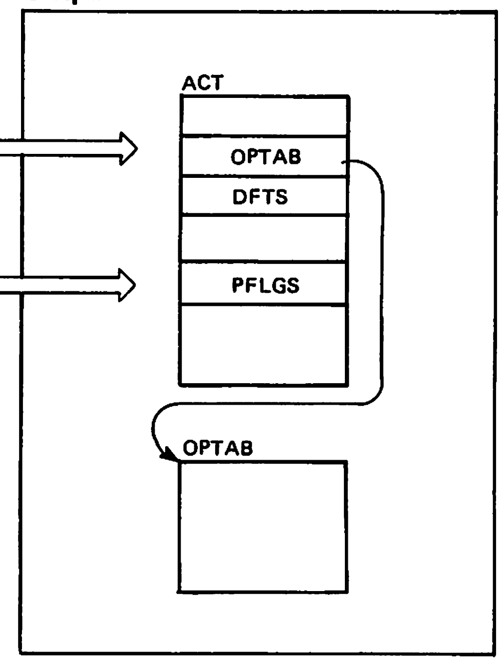


Diagram 7. Input Merge (ERBMFINP) (Part 2 of 4)

General Description

ERBMFINP is the first of the parse modules to receive control when a Monitor I (ZZ) session, a Background Monitor session, or a Monitor III session is either started or modified.

ERBMFINP controls the order in which the various option sources are parsed. Options are selected in the following order:

1 **Command Input Buffer (CIB)** — If the member option appears in the CIB, the members specified are processed *after* all other options in the CIB have been processed. See "Member Option."

2 **EXEC Card Parm Field** — The parm field of the RMF PROC EXEC card is parsed only when RMF is started and the ZZ session is to be started automatically. Thereafter, ACTEXEC will be 0. If the member option appears in the parm field, and no member option was previously parsed in the CIB, the members specified are processed *after* all other options in the EXEC card parm field have been processed. See "Member Option."

3 **Member Default** — If no member option was processed for an earlier source and the session command is "START," then the following members are processed for the session type indicated;

Session Type	*Default Member*
ZZ	ERBRMF00
BDM (background monitor)	ERBRMF01
Monitor III Data Gatherer	ERBRMF04
Monitor III Data Reporter	ERBRMF05

4 **Defaults** — Hard coded defaults are processed when starting a session.

5 **Operator Reply** — If "OPTIONS" (alias "OPTN") was selected from a prior source, or a parse error was detected (includes errors due to conflicting options detected by the Validity Checking Subroutine), the options selected will be listed on the operator terminal and operator intervention allowed. The reply will be parsed and any options specified will override those previously selected.

For each source, ERBMFINP calls ERBMFQOP to complete the parse for that source.

After all sources have been parsed, ERBMFINP returns with RC = 0 and the OCB chain built by ERBMFQOP (ACTOCB points to first OCB in chain).

Member Option — Up to five member IDs may be specified in the Member Option Entity. The members identified by the IDs are read and parsed according to the order specified (left to right).

Rules On Parsing a Particular Source

1 *Starting a Session, Except for Operator Reply* — Options are accepted from left to right. For example, if the following options are specified, "NOCPU, CHAN, CPU," the result will be "NOCPU, CHAN."

2 Modifying a Session or Operator Reply — The options are accepted from the right. The result of the example in 1 above would have the following result:

"CHAN, CPU"

Diagram 7. Input Merge (ERBMFINP) (Part 3 of 4)

No Diagram.

Extended Description continued on next page.

Diagram 7. Input Merge (ERBMFINP) (Part 4 of 4)

Extended Description	Module	Label

1 The environment and type of session command determine which option table to use and whether or not the appropriate defaults are required:

Command	*Session Type*	*ACTOPTAB*	*ACTDFTS*
START	ZZ	ERBTBZZS	ERBDFZZS
START	BACK.DISPLAY MON.	ERBTBBDM	ERBDFBDM
MODIFY	ZZ	ERBTBZZS	0
MODIFY	BACK.DISPLAY MON.	ERBTBBDM	0
START	Monitor III data gatherer	ERBTBM3G	ERBDFM3G
START	Monitor III data reporter	ERBTBM3R	ERBDFM3G
MODIFY	Monitor III data gatherer	ERBTBM3G	0
MODIFY	Monitor III data reporter	ERBTBM3R	0

2 If ACTCIBA ≠= 0 (set by ERBMFMFC when starting the ZZ session automatically, or by ERBMFCTL when RMF is modified), parsing of the CIB will be done. Note: This is the only source which already has ACTCHARP, ACTSRCP, and ACTENDP set.

3 If ACTEXEC ≠= 0 and "NOZZ" was *not* specified on the START RMFcommand, the parm field on the RMF PROC EXEC card is parsed.

4 If the session is being started (ACTSRTF = on) and no MEMBER option was parsed for a previous source (INMEMBF = off), the default member (see "General Description") is parsed.

Extended Description	Module	Label

5 Defaults appear in free text form in the module ERBOPTAB. ACTDFTS points to the default table to use.

6 Subroutine INPVALCK checks the final set of selected options for conflicts. The following conflicts are defined and will be flagged by message ERB301I which contains a numeric descriptor corresponding to the order following:

1. NOREPORT, NORECORD specified. RMF will change NOREPORT to REPORT (DEFER).
2. REPORT (DEFER), NOSTOP. RMF will change NOSTOP to STOP (interval time).
3. STOP (val1), INTERVAL (val2) where val1 < val2. RMF sets val1 = val2.

If conflicting options are detected, operator intervention will be forced.

7 Operator intervention allowed if either ACTOPIF = on or ACTERRF = on.

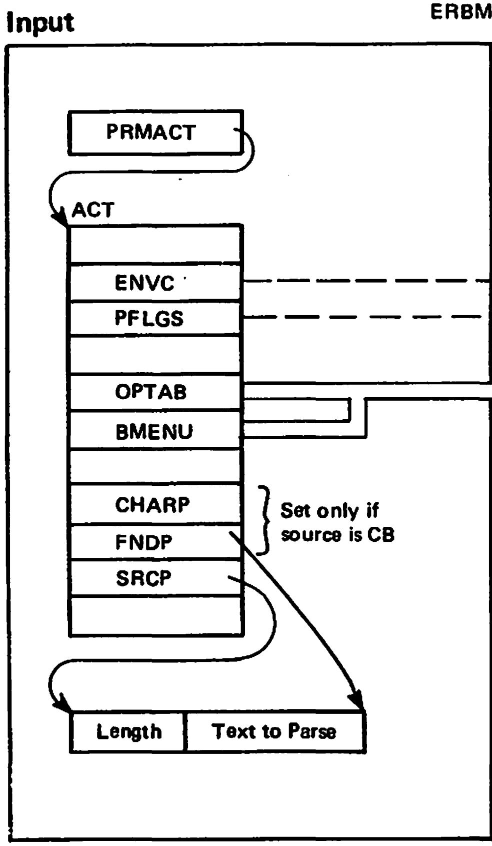
a. Call ERBLISTO to write ERB305I messages.

b.

c.

8

Diagram 8. Queue Options (ERBMFQOP) (Part 1 of 4)



ERBMFINP

Process

1 Initialize the parse variable according to the environment (ENVC) and command type (PFLGS).

2 Do this step until all text has been parsed.

a. Parse for an option.

b. If the option is not found (error occurred):

i. Suppress the writing of any more error messages.

ii. Back up and scan until a valid option is found.

iii. Call ERBMFQSV to list the text that was shipped in order to find a valid option.

c. If a valid option found:

ERBMFQSV

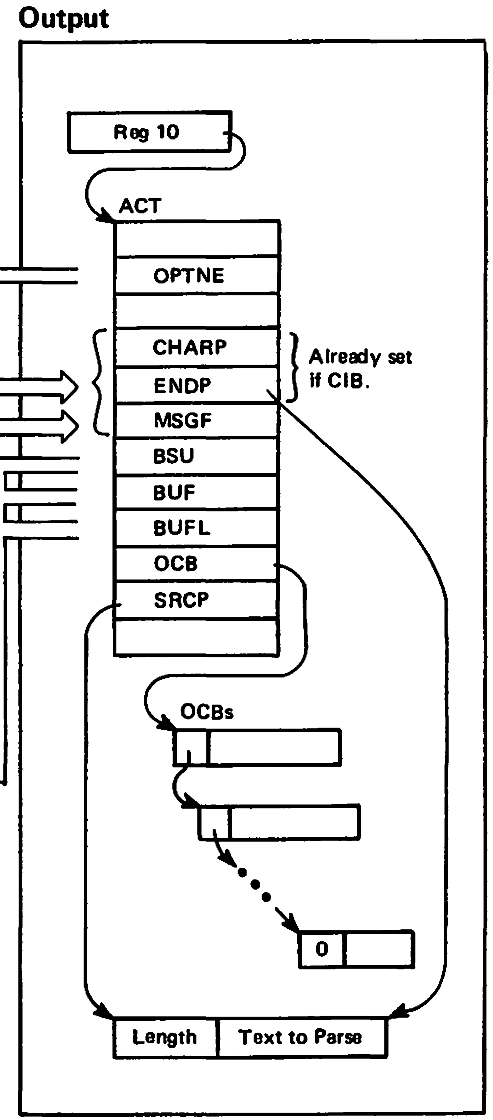


Diagram 8. Queue Options (ERBMFQOP) (Part 2 of 4)

Extended Description	Module	Label
ERBMFQOP drives the parsing of Monitor I and Monitor II options and builds an option control block (OCB) for each option, suboption, and trace variable found. The parsing of Monitor III options is driven by module ERB3CQOP. Any parsing routine commonly used by all monitors resides in ERB3CQOP. For example, the Monitor III REFRESH option has the same parsing requirements as the Monitor I and Monitor II INTERVAL option. The REFRESH and INTERVAL options have the same OPRTN number and are handled by a subroutine in the Monitor III module (ERB3CQOP).		
Common service routines (such as routines that write messages or build OCBs) used by all option parsing modules reside in module ERBMFQSV.		
1 Set ACTCHARP (the character currently being looked at) and ACTENDP (the end of the character string). These fields are already set if the CIB is the source of the variables. Also, indicate that error messages are to be written (ACTMSGF = '1'B).		
2 Scan until ACTCHARP = ACTENDP.		
a. Call ERBLEXAN to find the option. ERBLEXAN looks up the proposed option in tables passed to it (ERBOPTAB first; then, if this is a background monitor parse, ERBBMENU).	ERBLEXAN	
b. If ACTBSU=0 or ACTOPTNE=0 (return codes from ERBLEXAN), then:		
i) Set ACTMSGF='0'B.		
ii) Set ACTCHARP to point immediately after the last recognizable and acceptable text.		
iii) Call ERBMFQSV to list the text that was skipped.	ERBMFQSV	
c. If ACTOPTNE ≠ 0 (this is set by ERBLEXAN and points to the table entry for the option specified), then:		

Diagram 8. Queue Options (ERBMFQOP) (Part 3 of 4)

Input

ACT

ACTOPTNE

ERBOPTNE

OPRTN

Process

2 c. (continued)

i. Determine whether the option is acceptable.

ii. For a Monitor III option, call ERB3CQOP to parse the option. — ERB3CQOP

iii. For a Monitor I or II option, call the appropriate subroutine to parse the option.

iv. If an error occurred, set up to skip to the next option.

v. If no errors occurred, call ERBMFQSV to build an OCB. Prepare to find a delimiter and another option. — ERBMFQSV

vi. Go to step 2a. — step 2a.

3 All text has been parsed, return to caller (return code = 0).

ERBMFINP

Output

Reg 10

ACT

BSU

PFLGS

OCB

CHARP

NXTST

Diagram 8. Queue Options (ERBMFQOP) (Part 4 of 4)

Extended Description	Module	Label
2 (continued)		
c. (continued)		
i) If this is a START command and the OCB is already on the chain, set a flag so that this request is ignored (QOPIGNF='1'B).		
ii) If OPRTN is number 24 (the maximum number of routines to parse options in this module), the option is not for Monitor I or Monitor II. Call ERB3CQOP to parse the option.		
iii) If the option is for Monitor I or Monitor II, call the appropriate subroutine, based on OPRTN for the entry pointed to by ACTOPTNE, to parse the option entity.		
iv) Set ACTCHARP to the end of the last acceptable piece of text.		
v) Set ACTNXTST to find a delimiter and another option (see ERBLEXAN).		
vi) Go to Step 2a.		
3		

Diagram 9. Service for Queue Options (ERBMFQSV) (Part 1 of 2)

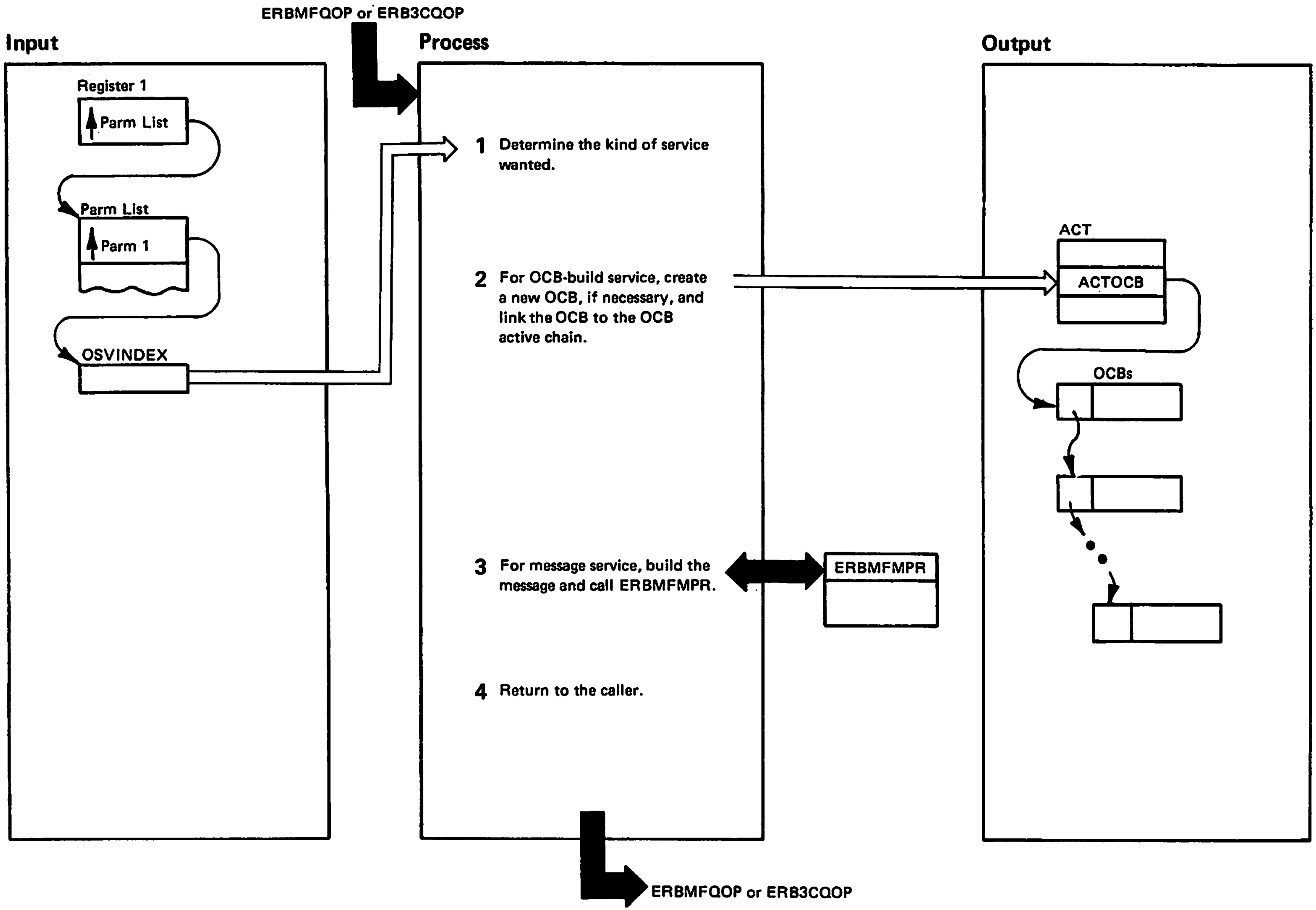


Diagram 9. Service for Queue Options (ERBMFQSV) (Part 2 of 2)

Extended Description	Module	Label
ERBMFQSV provides ERBMFQOP and ERB3CQOP with two types of service. ERBMFQSV either builds an OCB or writes a message.		
1 ERBMFQSV examines the input parameter OSVINDEX to determine the type of service requested.	ERBMFQSV	ERBMFQSV
2 If the input parameter OSVINDEX is zero, ERBMFQSV builds an OCB, if necessary, and links the OCB to the OCB active chain.		
3 If the input parameter OSVINDEX is not zero, OSVINDEX contains a message ID. ERBMFQSV calls ERBMFMPR to build and write the message.	ERBMFMPR	ERBMFMPR
4 Return to the caller, either ERBMFQOP or ERB3CQOP.	ERBMFQSV	ERBMFQSV

Diagram 10. Lexical Analyzer (ERBLEXAN) (Part 1 of 2)

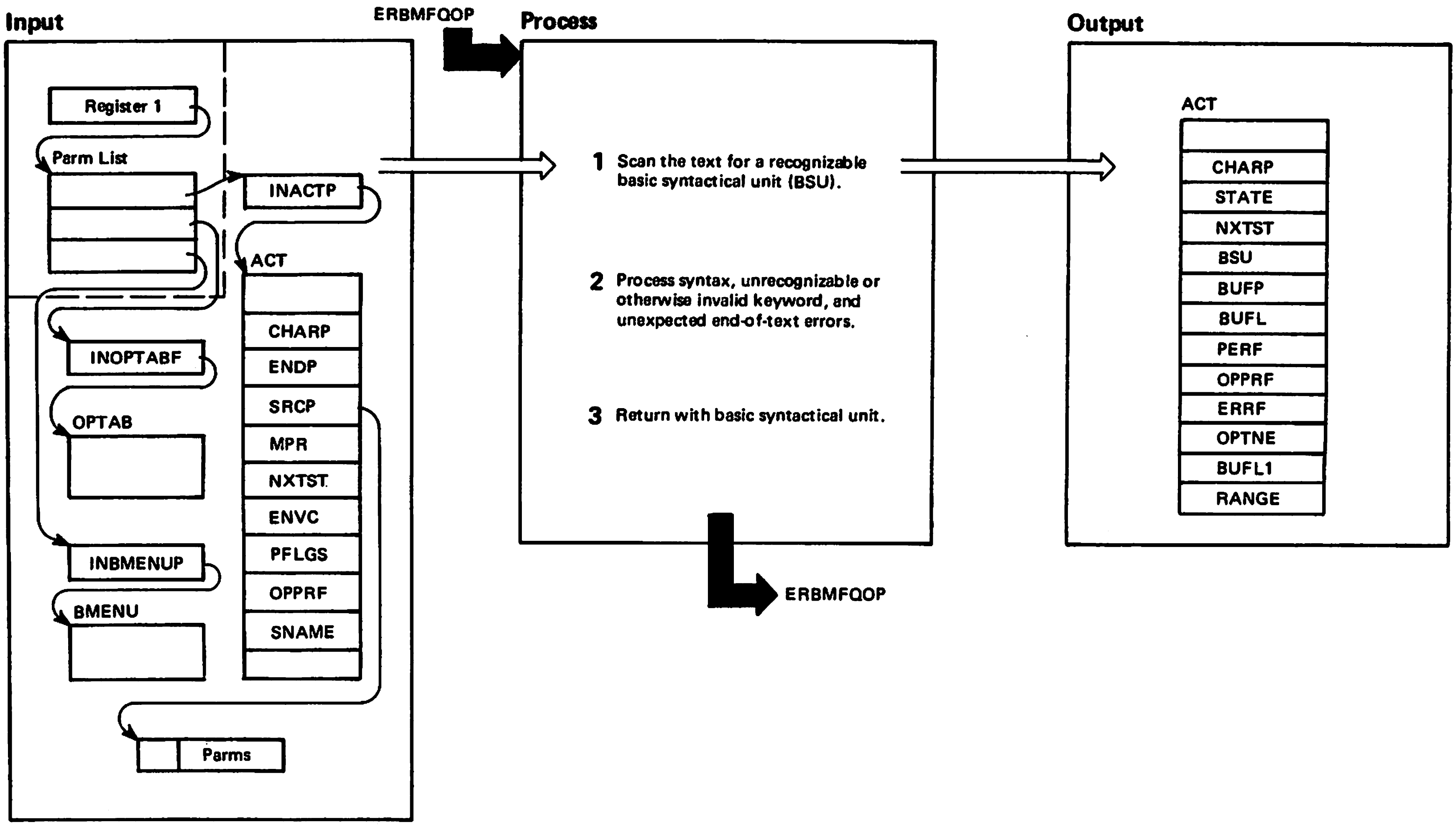


Diagram 10. Lexical Analyzer (ERBLEXAN) (Part 2 of 2)

General Description

ERBLEXAN scans the text passed (starting at ACTCHARP) and returns to caller with a recognizable token. This module also processes SYNTAX and invalid option errors.

Extended Description

1

ACTCHARP	=	PTR to current character
ACTENDP	=	PTR to end of parm string
ACTSRCP	=	PTR to parmstring-2
ACTMPR	=	PTR to ERBMFMPR (Message Processor)
ACTNXTST	=	Next State
ACTENVC	=	Environment Code Flags
ACTPFLGS	=	Processing Flags
ACTOPPRF	=	Option processing flag
ACTSNAME	=	Session name

Parsing is driven by a STATE/CLASS matrix, where the STATE indicates the acceptable characters at a particular phase (state) of the parse, and the class is a numeric descriptor for a unique group of characters or a single character. After the class is determined (TRT Table) for the character currently being looked at, ERBLEXAN indexes the STATE/CLASS matrix to obtain the next state (NXTST (ACTSTATE, RCLASS)) and an address of a routine to execute (ROUTINE (ACTSTATE, RCLASS)).

STATE-CLASS MATRIX

STATE	DESCRIPTION
1	SKIP BLANKS AND COMMENTS TO STING
2	COLLECT ALPHABETIC STRING
3	COLLECT NUMERIC STRING
4	COLLECT ALPHA-NUMERIC STRING (1ST CHAR NOT NUMBER)
5	COMMENT PROCESSING (/ FOUND, NEED *)
6	COMMENT PROCESSING (NEED FINAL *)
7	COMMENT PROCESSING (NEED FINAL /)
8	COLLECT VALUE H, VALUE M, OK VALUE S
9	NEED DELIMITER, THEN GO FIND ANOTHER OPTION
10	COLLECT ALPHA-NUMERIC STRING (1ST CHAR A NUMBER)
11	BACKGROUND MONITOR OPTION FOUND, NEED '(' OR DELIMITER
12	OPTION FOUND, NEED '(' THEN SUBOPTION
13	HAVE SINGLE SUBOPTION, NEED ')' THEN DELIMITER
14	HAVE SUBOPTION, NEED ')' OR DELIMITIER-SUBOPTION
15	BACKGROUND MONITOR OPTION PARM FIELD PROCESSING, NEED ')'

CLASS	CHARACTERS
1	$, @, #, AND ALL LETTERS EXCEPT M,H,AND S
2	LETTERS H,M,AND S
3	NUMBER
4	(
5	)
6	COMMA
7	BLANK
8	/
9	*
10	.
11	EVERYTHING ELSE
12	NONE – USED FOR END-OF-TEXT PROCESSING

The routine chosen will perform the necessary processing, whether normal or error, for the current CLASS of character for the given STATE of parse. The following is a list of the routines and functions:

Extended Description (Continued)

LEXRTN1 – Syntax error at ACTCHARP, return, ACTBSU = BSUERROR

LEXRTN2 – Begin collecting a string of characters, continue

LEXRTN3 – Continue

LEXRTN4 – End of alphabetic string collection, keyword look up if necessary, return

ACTBSU = {
BSUALPHA – alphabetic string
BSUOP – keyword is option in OPTAB
BSUBMOP – keyword is option in BMENU
}

Also determines whether character string is an unrecognizable option or otherwise invalid (calls LEXMPR).

Sets ACTBUFP = Beginning of char. string.
Sets ACTBUFL = Length of char. string.
Sets ACTOPTNE = PTR to either option table entry of keyword found in OPTAB or background menu entry if found in BMENU.

LEXRTN5 – Continue

LEXRTN6 – Colon found, set ACTRANGE = '1'B, continue

LEXRTN7 – valueH, valueM, or valueS discovered, set ACTBSU, continue

ACTBSU = {BSUVALH, BSUVALM, BSUVALS}

LEXRTN8 – First left parentheses delimiting background menu option was found, set paren counter = 1, continue

LEXRTN9 – Char. string is now alphanumeric with first character a number, ACTBSU = BSU#ALPH, continue

LEXRTN10 – Comma delimiter found (continue)

LEXRTN11 – All delimiters, set ACTOPPRF='0'B, continue

LEXRTN12 – Begin a comment, save return state, continue

LEXRTN13 – End of Text Processing, determine whether end is unexpected (call LEXMPR), return

LEXRTN14 – End of comment, return to state saved at LEXRTN12

LEXRTN15 – String is now alphanumeric, first character not numeric, ACTBSU = BSUALPH#, continue

LEXRTN16 – Comma found (return)

LEXRTN17 – All delimiters, return – ACTBSU = BSUDELIM

LEXRTN18 – Right parentheses found, ACTBSU=BSURP, return

LEXRTN19 – Syntax error, something missing (CALL LEXMPR), continue

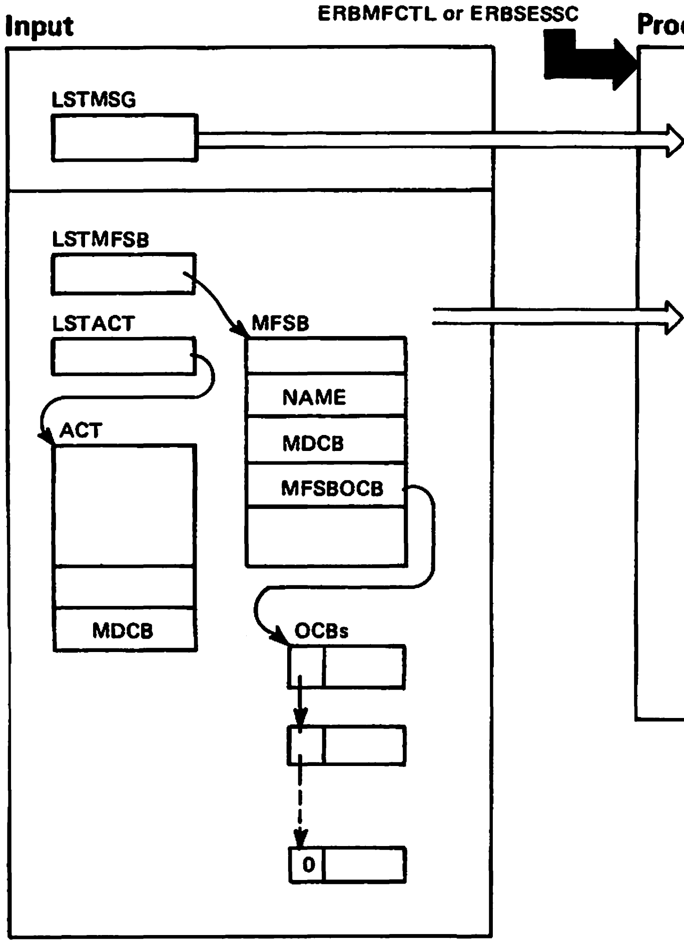
LEXRTN20 – Embedded left parentheses, increment paren counter, continue

LEXRTN21 – Right parenthesis found for background monitor parm field, decrement paren counter. If >0 continue, else return with ACTBSU = BSURP.

2 Subroutine LEXMPR calls ERBMFMPR to write one of the following error messages: ERB300I, ERB219I, ERB223I, ERB220I, ERB221I.
Set ACTERRF='1'B.

3

Diagram 11. List RMF Options (ERBLISTO) (Part 1 of 2)



Process

1 Set up the header message for the message indicated by LSTMSG.

2 For each OCB (option control block) on the chain, build the interface for ERBMFMPR and call ERBMFMPR to write the appropriate message.

3 Return.

To Caller

Diagram 11. List RMF Options (ERBLISTO) (Part 2 of 2)

Extended Description	Module	Label
1		
2 ERBLISTO calls ERBMFMPR to write message ERB103I or ERB305I to list the options designated by the OCBs hung off the OCB chain pointer in the MFSB.		
3 Return code = 0.		

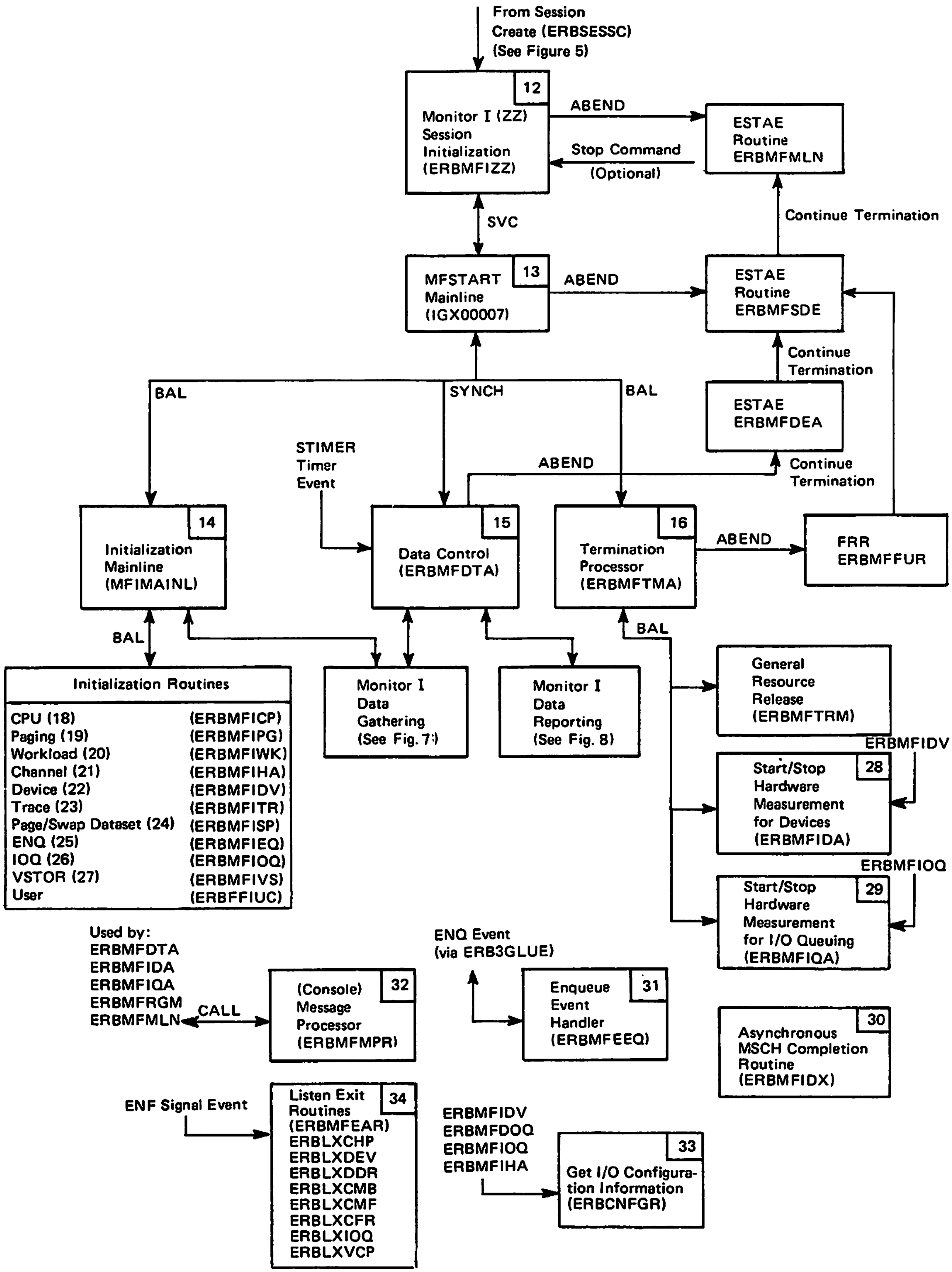
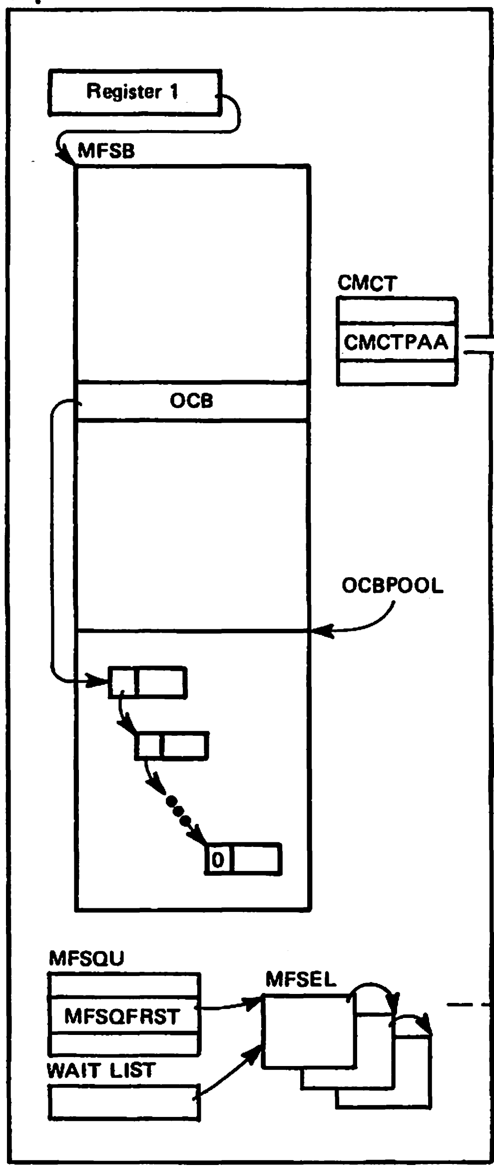


Figure 6. Monitor I Control Overview

Diagram 12. Monitor I (ZZ) Session Initialization (ERBMFIZZ) (Part 1 of 4)



ERBSESSC via ATTACH

Process

1 Serialize the ZZ session processing.

2 Initialize the session control blocks.

3 Map the options into the ZZ session control blocks.

4 Establish recovery (ESTAE).

5 Start the RMF measurements.

6 Cancel recovery unless a SET command was issued.

7 If a SET command was issued, save return code, set indicator, and continue.

8 If the return code from step 5 = 0:

a. Wait for all subtasks to complete.

b. Release storage for terminated subtasks.

c. Go to step 11.

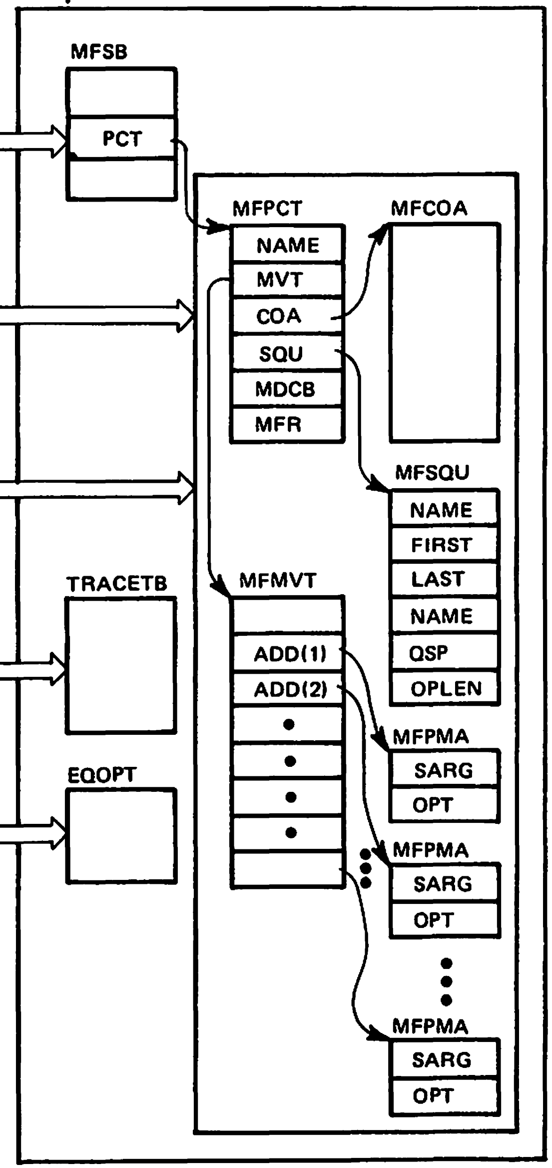


Diagram 12. Monitor I (ZZ) Session Initialization (ERBMFIZZ) (Part 2 of 4)

Extended Description	Module	Label
ERBMFIZZ initiates the collection of system activity measurements and terminates after all reports on these activities have been formatted. This is accomplished by calling the MFSTART SVC with the initialized control blocks that indicate the measurement and control options.		
After the MFSTART SVC returns, ERBMFIZZ detaches any remaining report generator subtasks and releases subtask control storage.		
If a SET command was issued (the return code from MFSTART is 12), reinitialize for the beginning of a new ZZ session.		
If a modify was issued for the ZZ session, ERBMFCTL will post MFSBMECB. ERBMFDTA will then force the termination of the active ZZ session and return to ERBMFIZZ. ERBMFIZZ will wait for all report generators' subtasks to complete and then reinitialize for the beginning of a new ZZ session using the modified options.		
If a listen exit could not be activated by MFIMAINL (MFSTART return code=16), the RMF appropriate option is disabled (OCBNOXF=1), MFSBNENF is set on, and the session terminated. ERBMFIZZ will then reinitialize a new ZZ session using the changed options.		
If ERBMFIQA could not start the generation of hardware measurements (MFSTART return code=16), the RMF I/O queuing option is disabled (OCBNOXF=1), MFSBNHMA is set on, and the session is terminated. ERBMFIZZ then reinitializes a new ZZ session using the changed options.		
Note: ERBMFIZZ posts MFSBMODC before issuing the MFSTART SVC. ERBMFCTL will write a message indicating that the session is modified when this happens.		
1 ENQUEUE on SYSZRBZZ ACTIVE. If ENQ is not successful, abend with user code 1200 and response code 4.		

Extended Description	Module	Label
2 All control blocks are from automatic storage.		
3 ERBMFIZZ has a separate subroutine for each ZZ option that sets the flags and fields for that option in the ZZ session control blocks.		
If the CHAN option is selected, the CMCTPAA bit is tested to determine if the SRM store channel path measurement facility is active. If no channel path status data is available (CMCTPAA=0), issues the message ERB264I and ERB260I and deactivate the CHAN option (OCBNOXF='1').		
4 Issue an ESTAE macro instruction to establish linkage to the error recovery routine (ERBMFMLN).	ERBMFMLN	
5 Issue MFSTART SVC 109 (code=7). This starts the measurements set by input merge control in tables MFCOA and MFPMA (pointed to by MFMVT). Control passes from MFSTART to data control (ERBMFDTA) before returning to ERBMFMFC. This step is also the entry point for a retry requested by the recovery routine (ERBMFMLN).		
6 Unless a SET command was issued, issue an ESTAE macro instruction to deactivate the error recovery linkage (ERBMFMLN).		
7 If a SET command was issued (the return code from MFSTART is 12), save the return code in a buffer (RCBFR), set the return code (MFSVCCOD) to zero, and go to Step 8.		
8 Insure all resources are released.		

Diagram 12. Monitor I (ZZ) Session Initialization (ERBMFIZZ) (Part 3 of 4)

Process

9 If Step 5 return code = 4, call ERBMFMPR to write ERB202I message (No ZZ measurements selected) and go to Step 10.

10 If Step 5 return code ≠ 0, ≠ 4 and ≠ 16, ABEND with USER = 1200.

11 If session termination was caused by MODIFY session command, go to Step 3.

12 If session termination occurred either because MFIMAINL could not allocate an ENF listen exit, or because ERBMFIQA could not start the generation of hardware measurements, go to step 3.

13 If a SET command was issued, go to Step 5.

14 Return.

ERBMFCTL WAITS for Completion

Diagram 12. Monitor I (ZZ) Session Initialization (ERBMFIZZ) (Part 4 of 4)

Extended Description	Module	Label
9 If Step 5 return code=4, call ERBMFMPR to write message ERB202I (No ZZ measurements selected) and go to Step 10.		
10 Reason code 8 is provided with ABEND 1200.		
MFSTART failed because the ENQ name SYSZRBZZ. ACTIVE was not held by the calling task or because ZZ session initialization has already been performed.		
11 If MODIFY was issued, MFSBMECB ≠ 0 is set up to reinitialize new options.		
12 If RCBFR=16, go to Step 3 to reinitialize RMF with changed options.		
13 If RCBFR=12, go to Step 5 to reinitialize RMF using the same parameter list that was used before the SET command was issued.		
14 Return codes : 0 – Successful, or no measurements selected.		

Diagram 13. MFSTART Mainline (IGX00007) (Part 1 of 2)

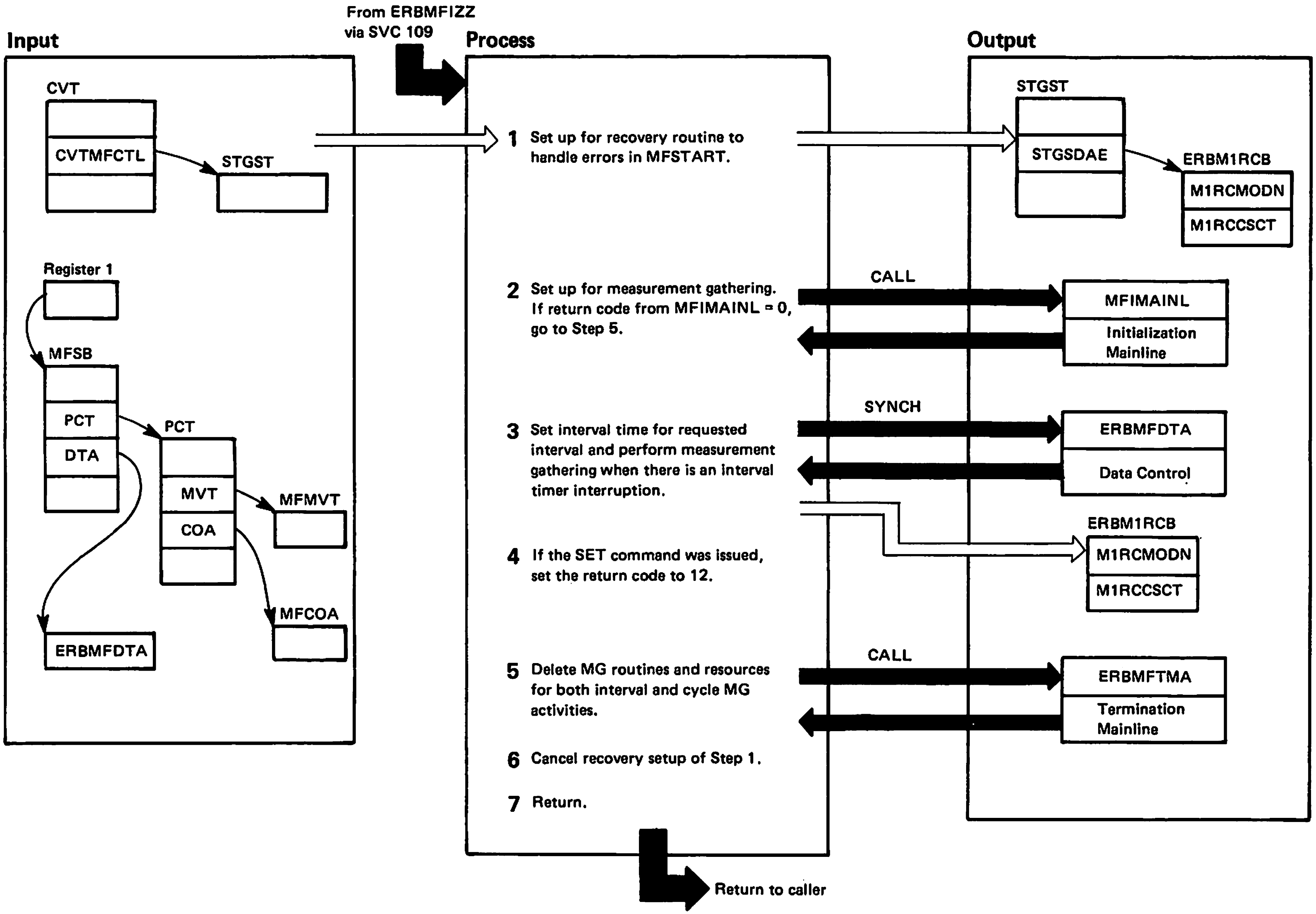


Diagram 13. MFSTART Mainline (IGX00007) (Part 2 of 2)

Extened Description	Module	Label
The MFSTART Mainline (IGX00007) processor controls the initialization and termination of routines that perform Monitor I functions.	IGX00007	
MFSTART controls the initialization of all supervisor-state control tables and global resources for RMF measurements, obtains initial measurement data wraparound values and time stamps, allows collection of subsequent measurements under error recovery control, and controls the deallocation of all RMF supervisor state resources, including global resources.		
MFSTART Mainline establishes an area for itself and other modules to use to set footprints for use during recovery processing. It obtains storage for this area (ERBM1RCB) and stores its address in STGSDAE. MFSTART Mainline then places its own module/CSECT name in the area. Each successive module called during Monitor I processing places its own module/CSECT name (or footprint) in the area, thus enabling recovery routines to identify the module that was active when an error occurred.		
1 Issue an ESTAE macro instruction to provide entry to routine ERBMFSDE, which receives control in event of RMF errors.	IGX00007 ERBMFSDE	
2 Call the Initialization routine (MFIMAINL), which, in turn, calls other initialization routines.		MFIMAINL
3 Use SYNCH macro instruction to change to problem state and to transfer control to the Data Control routine (ERBMFDTA), which sets the interval timer and initiates measurement gathering after each interval.	ERBMFDTA	
4 If a SET command was issued (SRM has posted the MFSBSECB), set the return code to 12.		
5 After the last interval, Data Control returns control to MFSTART Mainline, which calls Termination Mainline (ERBMFTMA).	ERBMFTMA	
6 MFSTART Mainline cancels the ESTAE routine entry.		
7 Returns to the caller with one of the following return codes:		

Extended Description	Module	Label
00 – Requested options were valid, initial values have been set and the linkages to the data collection routines have been established.		
04 – One or more invalid input options were detected or no input measurement options were detected. RMF measurements were not initialized and control was not given to ERBMFDTA (data control) to collect RMF system activity measurements.		
08 – The ENQ name SYSZRBZZ.ACTIVE was not available and not held by the calling task, or RMF initialization has already been performed for this task. The SYNCH was not taken for measurement collection. No RMF initialization was done.		
12 – A SRM SET command was issued. The SRM posts MFSBSECB. This caused ERBMFDTA to come out of the wait and return to MFSTART. The ECB is checked for posting, then return code 12 is set.		
16 – A request issued to ENF to activate a listen exit was unsuccessful, or ERBMFIQA could not start the generation of hardware measurements. The option in error is set inactive. ERBMFIZZ will restart the session.		
ERBMFIZZ restarts the session.		

Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 1 of 12)

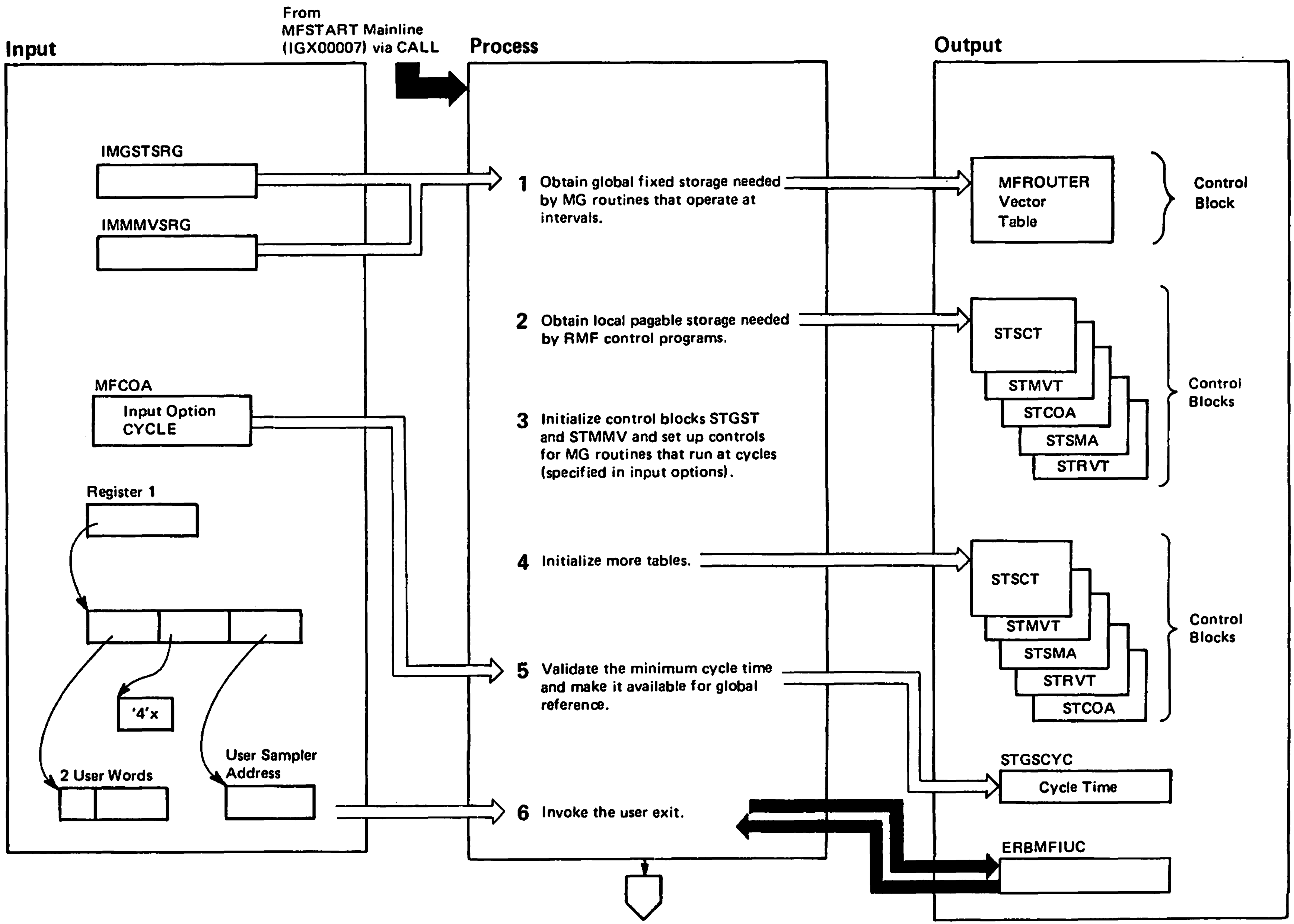


Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 2 of 12)

Extended Description	Module	Label
The Initialization Mainline (MFIMAINL) procedure controls the allocation of space for and the initialization of control blocks. It also calls routines whose purposes are to initialize different functions essential to measurement gathering (MG). Finally, it issues the MFDATA SVC to collect initial values of requested measurements.	IGX00007	MFIMAINL
1 MFIMAINL uses the GETMAIN macro instruction to obtain storage for the MFROUTER (control routine for sample collecting routines) Vector Table (STMMV).	IGX00007	MFIMAINL
2 MFIMAINL uses the GETMAIN macro instruction to obtain storage for the Supervisor Control Table (STSCT), Measurement Vector Table (STMVT), the Common Option Area (STCOA), Supervisor Measurement Area (STSMA), and the Resource Vector Table (STRVT).	IGX00007	
3 MFIMAINL places initial values into the control blocks for which space was obtained in Step 1 and into the global supervisor table (STGST).	IGX00007	
4 MFIMAINL places initial values into the control blocks for which space was obtained in step 2.	IGX00007	
5 The time specified by the cycle input option must not be less than 50 milliseconds.	IGX00007	
6 Store the module/CSECT name of the user exit for ZZ initialization in ERBM1RCB, then call the user exit. On return, restore the contents of ERBM1RCB.	ERBMFIUC	

Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 3 of 12)

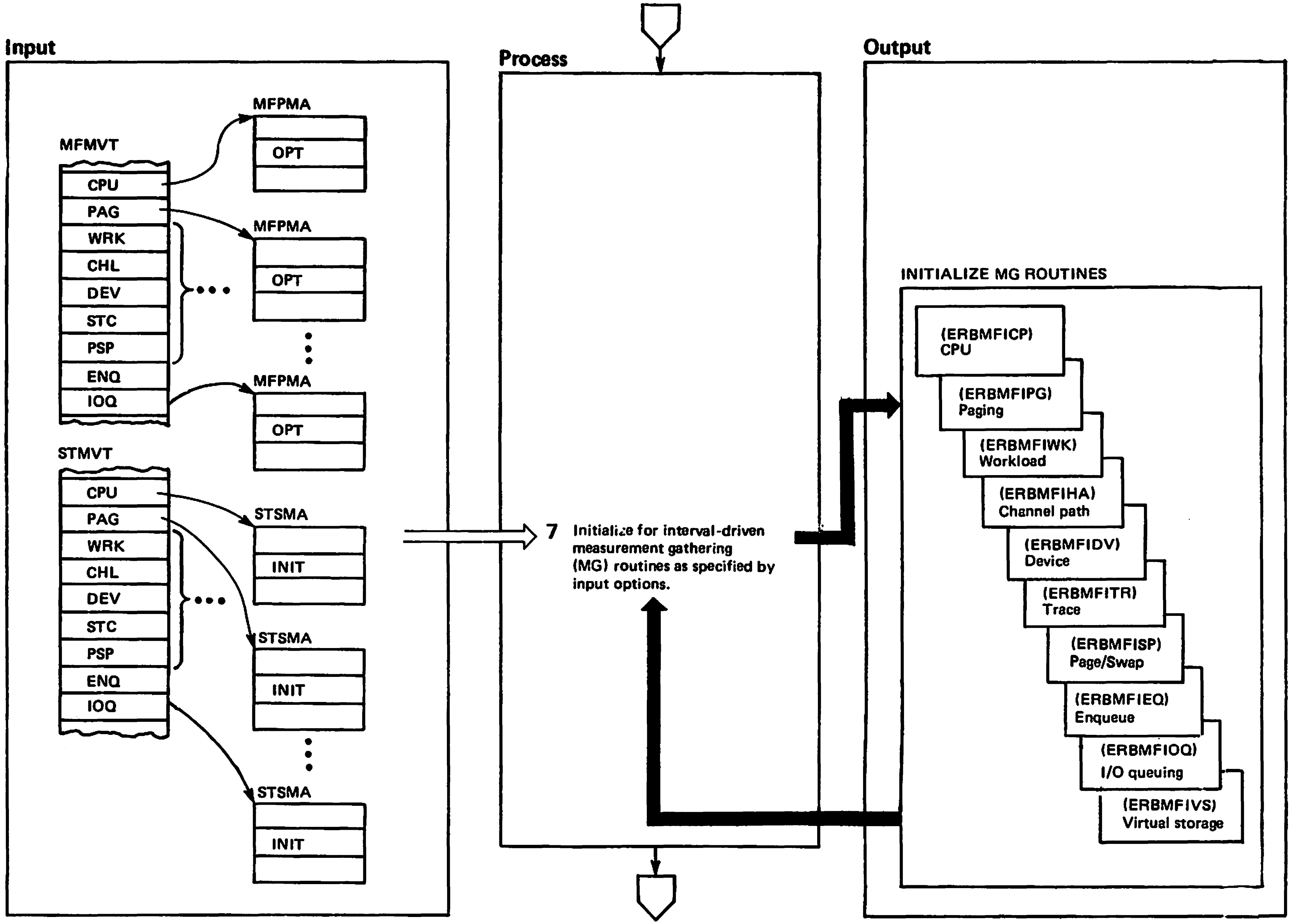


Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 4 of 12)

Extended Description	Module	Label
7 MFIMAINL calls the routines that initialize the MG routines. Only those MG routines required for the requested kinds of reports are called. For example, if CPU is the only requested report, then ERBMFICP is the only MG routine called. For recovery purposes, MFIMAINL stores in ERBM1RCB the module/CSECT name of any called MG routine, then restores its own module/CSECT name upon return.	ERBMFICP ERBMFIPG ERBMFIWK ERBMFIHA ERBMFIDV ERBMFITR ERBMFISP ERBMFIEQ ERBMFIOQ ERBMFIVS	

Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 5 of 12)

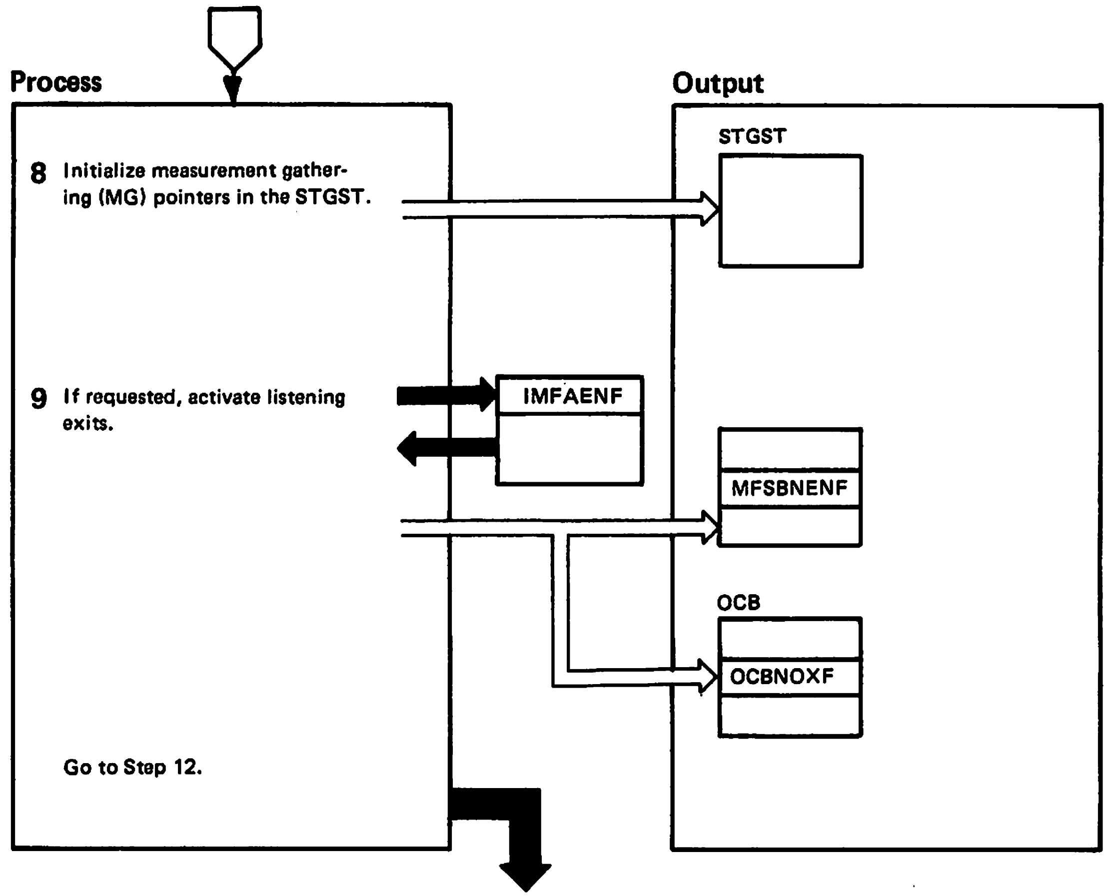


Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 6 of 12)

Extended Description	Module	Label
8 The list of data pointers, together with their corresponding option flags, is used by the listen exit routines and by Monitor II routines. If ERBMFIQA could not start the generation of hardware measurements (MFSBNHMA is on), calls subroutine IMRSIOQ to reset the I/O queuing option, sets a return code of 16, and goes to step 12 to return to the caller.		IMRSIOQ
9 Call subroutine IMFAENF to activate the appropriate listen exit routines (ERBMFEAR) for device, channel, and IOQ measurement options. If the IMFAENF return code is not zero, and the device or channel listening exit could not be activated, deactivate the measurement option (OCBNOXF=1). Then indicate session restart is necessary (MFSBNENF) because of ENF failure, and call		IMFAENF
ERBMFMPR to issue message ERB260I. If ERBMFIQA could not activate generation of hardware measurements, calls subroutine IMRSIOQ to reset the I/O queuing option.	ERBMFMPR	IMRSIOQ

Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 7 of 12)

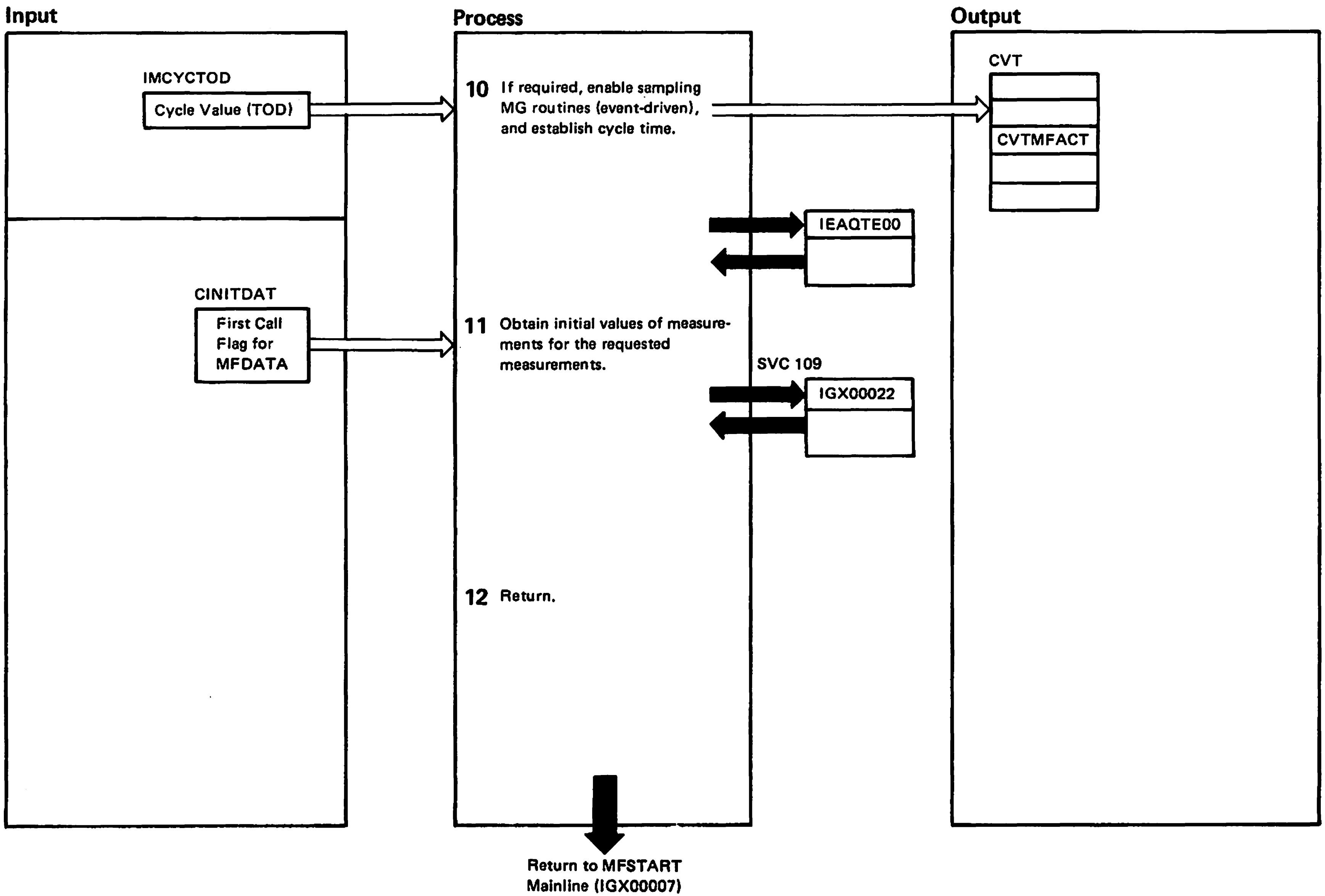


Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 8 of 12)

Extended Description	Module	Label
10 If a measurement other than CPU or workload is requested, MFIMAINL sets a flag in the communications vector table (CVT) in the field CVTMFACT. MFIMAINL also puts the time of the next sample into the RMF time queue element (TQE).	IGX00007	
Before calling routine IEAQTE00 to enqueue the TQE on the timer queue, MFIMAINL obtains the dispatcher lock and establishes a functional recovery routine (FRR) exit, after setting the TQE, these sections are reversed.	IEAQTE00	
11 MFIMAINL issues the MFDATA SVC (SVC 109), code 22, to collect data as requested by input options. This first call to each is indicated as the initial call and results in taking initial values against which later values are compared.	IGX00022	
For recovery purposes, MFIMAINL stores in ERBM1RCB the module/CSECT name of any called routine, then restores its own module/CSECT name upon return.		
MFIMAINL calls module ERBMFMPR to issue message ERB100I, indicating that the ZZ session is active.		
12 Return to caller with one of the following return codes:		
00 - Successful		
04 - One or more invalid input options were detected or no measurement options were specified. RMF measurements were not initialized and initial values were not obtained.		
08 - The ENQ name SYSZRBZZ.ACTIVE was not available and not held by the calling task, or RMF initialization has already been performed for this task or another task. No RMF initialization was done.		
16 - A request issued to ENF to activate a listen exit was unsuccessful, or, for 4831 environment, could not activate measurements. The option in error is deactivated and ERBMFIZZ will restart the session.		

Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 9 of 12)

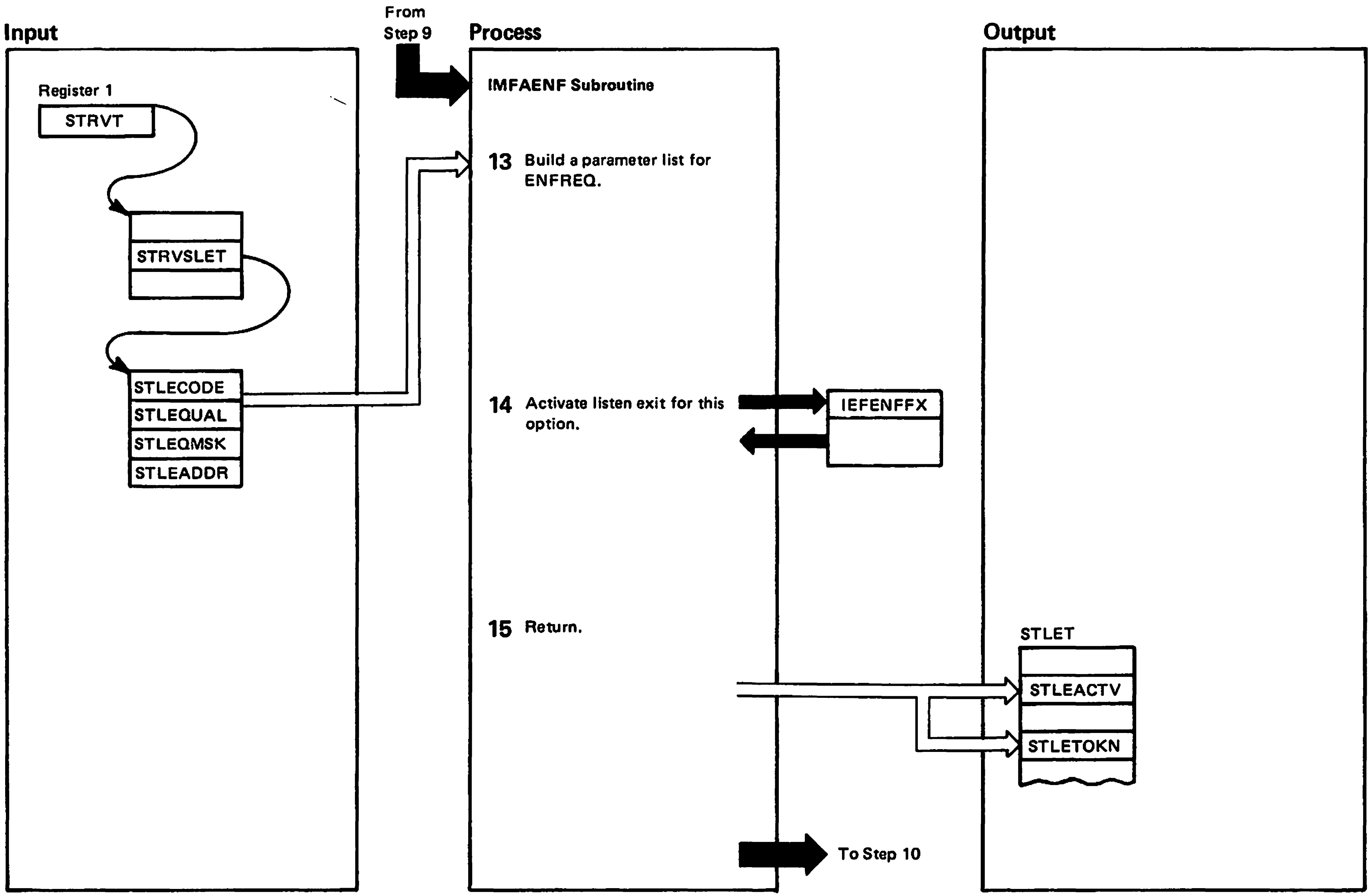


Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 10 of 12)

Extended Description	Module	Label
13 Build the parameter list for ENFREO macro instruction.		
14 Issue the ENFREQ macro instruction for each entry in STLET to activate listen exit for this option. On successful return from ENF, save the token (needed for deactivation) and mark this STLET entry active (STLEACTV = 1). If the request was not successful, issue an error message.	IEFENFFX	
15 Return to caller with one of the following return codes: 00 – Listen exits activated 04 – Error return code from ENF. At least one exit was not activated.		

Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 11 of 12)

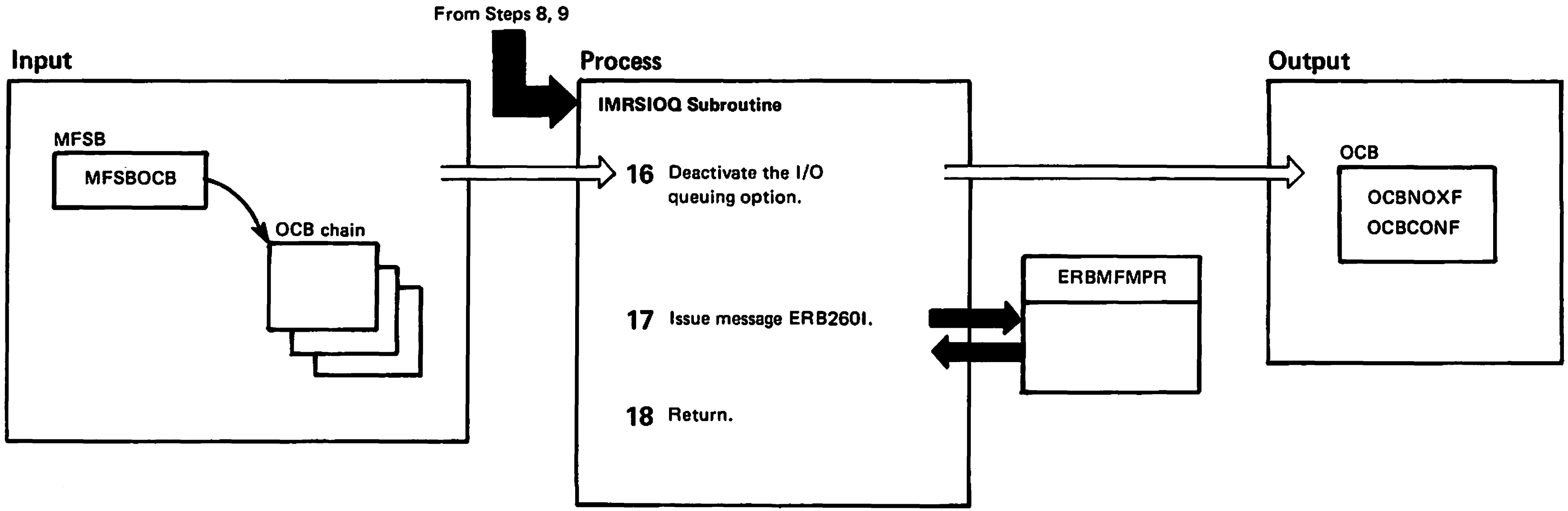


Diagram 14. Initialization Mainline (MFIMAINL) Subroutine of IGX00007 (Part 12 of 12)

Extended Description	Module	Label
16 Loop through the OCB chain to find the I/O queuing option OCB, and deactivate the I/O queuing option.		
17 Call module ERBMFMPR to issue message ERB260I, indicating the termination of I/O queuing activity.	ERBMFMPR	
18 Return to the caller.		

Diagram 15. Data Control (ERBMFDTA) (Part 1 of 6)

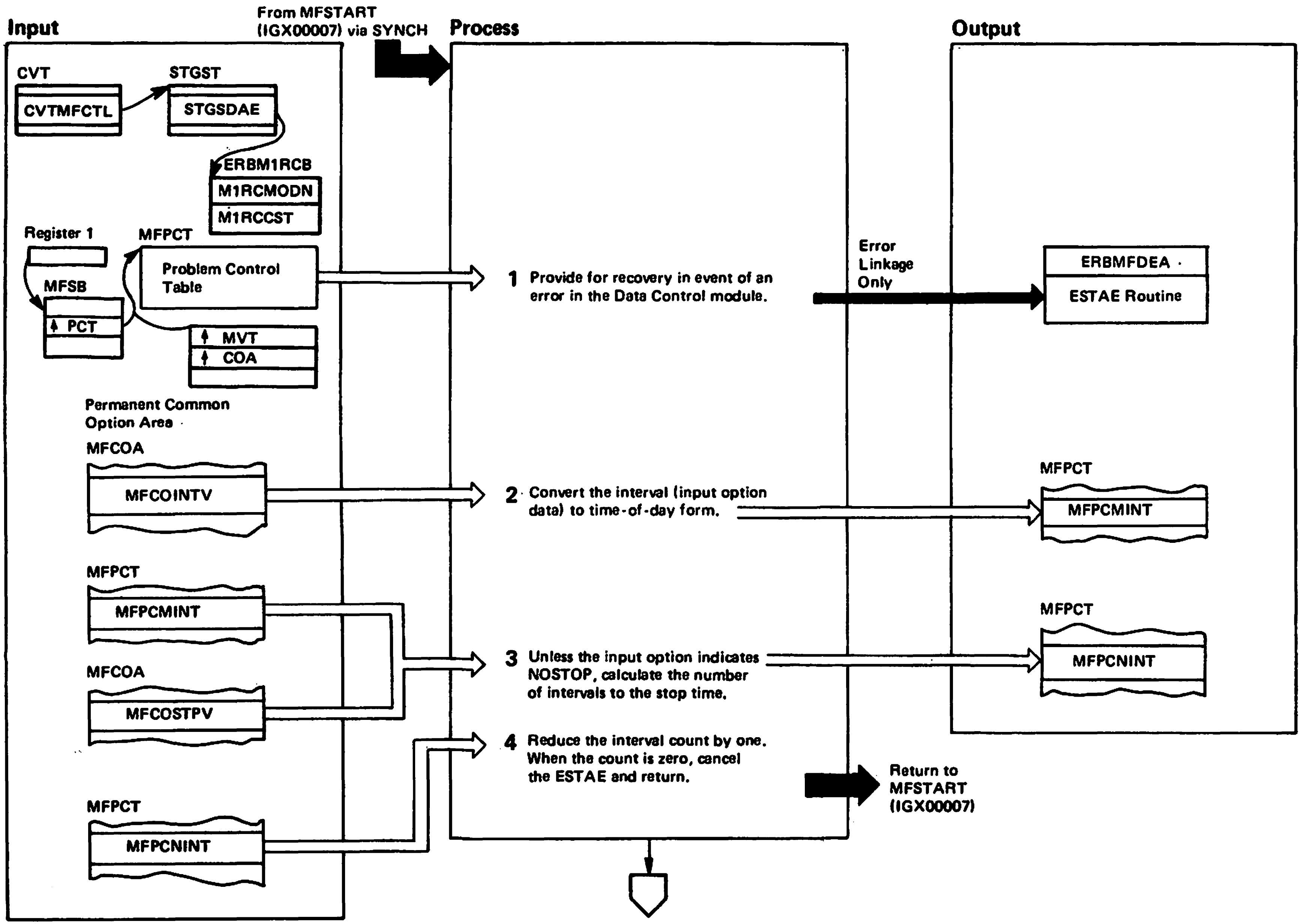


Diagram 15. Data Control (ERBMFDTA) (Part 2 of 6)

Extended Description	Module	Label
Data Control (ERBMFDTA) is executed in problem state in response to a SYNCH macro instruction issued by the MFSTART module. This change from supervisor state in MFSTART represents the entry into the main measurement gathering operations, which are controlled from the Data Control Module. Control includes establishing the interval of measurement gathering, as specified by an input option, and the queueing of report generation subtasks if real time reporting was requested. In addition, Data Control performs a number of event control block and storage control functions. For recovery purposes, Data Control switches briefly to supervisor state to set its own footprint (or the footprint of any module that it calls) in ERBM1RCB.	ERBMFDTA	
1 Data Control issues the MODESET macro instruction to switch to key zero and supervisor state, saves its module/CSECT name (footprint) in ERBM1RCB, and issues MODESET again to return to problem state. Data Control then establishes its ESTAE routines.	ERBMFDEA	
2 Interval time is entered in minutes. This time is converted to binary and placed in a full word of storage. When the low order bit in the full word is "on" and all other bits are "off", the time equals .01 seconds.	ERBMFDTA	
3 A stop time (input option) is specified or NOSTOP is specified. If NOSTOP is specified, the stop command is used to stop RMF operation. If a stop value is given, the amount of time from the current time until the stop time is divided by the interval length to obtain the number of intervals.	ERBMFDTA	
4 Data Control reduces the number of such intervals each time through this code. When this interval count is zero, RMF measurements are ended.	ERBMFDTA	

Diagram 15. Data Control (ERBMFDTA) (Part 3 of 6)

Input

MFPCT
MFPCELAD ≠ 0

Operator STOP Command
STOP Command ECB

SARG Subtask Ended
MFSEL
MFSESECB

STIMER Time Ended
STIMER ECB
DTSMECB

Process

5 Set interval timer to alert Data Control when the end of the current interval is reached.

6 If a previous list of event control blocks exists, free that storage, set up storage for a new ECB list, and set ECB addresses into new ECB list.

7 Wait for the posting of one of the following events (Step 8, 9, 10 or 11) in an event control block (ECB).

8 If the ECB for an operator-entered STOP command is posted, cause measurement gathering for this interval. If reports are to be printed, attach report generation subtasks in accord with input options. Cancel the ESTAE, and end measurement gathering by returning.

SVC 109
IGX00022
Start measurement gathering routines

ERBMFRGM
Report Generator

Return to MFSTART (IGX00007)

9 If the ECB for the end of a report generator subtask is posted, indicate its completion and free its main storage.

10 If the ECB for the end of the current interval is posted, cause measurement gathering for this interval and, if reports are to be printed, attach report generation subtasks in accord with input options.

Output

(SVC 109)
IGX00022
Start MG Routines

(ATTACH)
ERBMFRGM
Report Generator

Diagram 15. Data Control (ERBMFDTA) (Part 4 of 6)

Extended Description	Module	Label
5 The routine sets the STIMER macro instruction for the length of the current interval and compensates for any stop during the interval.	ERBMFDTA	
6 It uses one FREEMAIN macro instruction to free storage of any existing event control blocks (ECBs). Then the routine uses GETMAIN to obtain storage for pointers to ECBs: one ECB for the STOP command, one for the STIMER alert, and one for each report generation (SARG) subtask.	ERBMFDTA	
7 One of four conditions has occurred when an ECB is posted:	ERBMFDTA	
a) The operator has issued a stop command. If so, create short interval data, and end measurements. Return to caller of Data Control.		
b) A report generator subtask has ended. If so, detach the subtask, and dequeue its subtask element (SEL) from the subtask queue (SQU).		
c) The STIMER interval has been reached (the current interval has ended). If so, issue an MFDATA SVC to cause measurement gathering for this interval and attach a report generation subtask unless no report of these measurements was requested. Build a (SARG) subtask queue element (MFSQU) for the subtask.		
d) The operator has issued a SET command to reset the IPS, OPT, or installation control specification. Create data for a short interval report and terminate measurement gathering. Return to the calling routine.		

Extended Description	Module	Label
For recovery purposes, Data Control stores (in ERBM1RCB) the module/CSECT name of whichever routine it calls and restores its own footprint upon return. Data Control must switch to supervisor state to store to ERBM1RCB.		
8 An EXTRACT macro instruction is used to obtain the command input buffer (CIB) address of the STOP.	ERBMFDTA	
A short interval results when the STOP command is issued. The MFDATA SVC controls the collection of requested measurement data.	IGX00022	
Report generation subtasks are called by attaching the Report Generator control (ERBMFRGM).	ERBMFRGM	
9 Data Control issues a DETACH macro instruction to remove a completed subtask and then shortens the subtask queue. The subtask's main storage (its element subpool space) is freed by means of a FREEMAIN macro instruction.	ERBMFDTA	
10 The MFDATA SVC controls the collection of requested measurement data.	IGX00022	
Report generation subtasks are called by attaching the Report Generator control (ERBMFRGM).	ERBMFRGM	

Diagram 15. Data Control (ERBMFDTA) (Part 5 of 6)

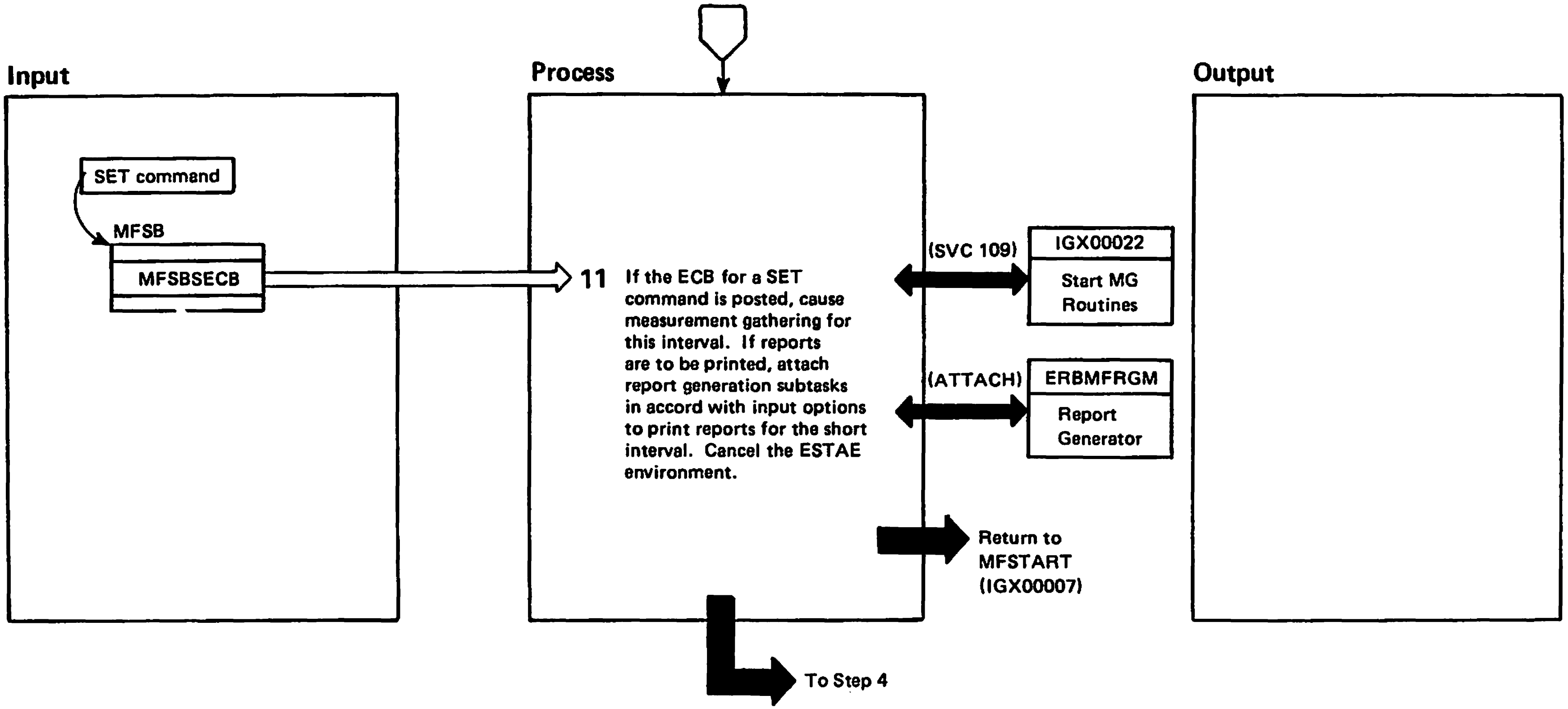


Diagram 15. Data Control (ERBMFDTA) (Part 6 of 6)

Extended Description	Module	Label
11 If the ECB for a SET command (MFSBSECB) is posted, generate the requested output for the short interval. The MFDATA SVC controls the collection of requested measurement data. Report generation subtasks are called by attaching the Report Generator conrol (ERBMFRGM).	IGX00022 ERBMFRGM	

Diagram 16. Termination Processor (ERBMFTMA) (Part 1 of 2)

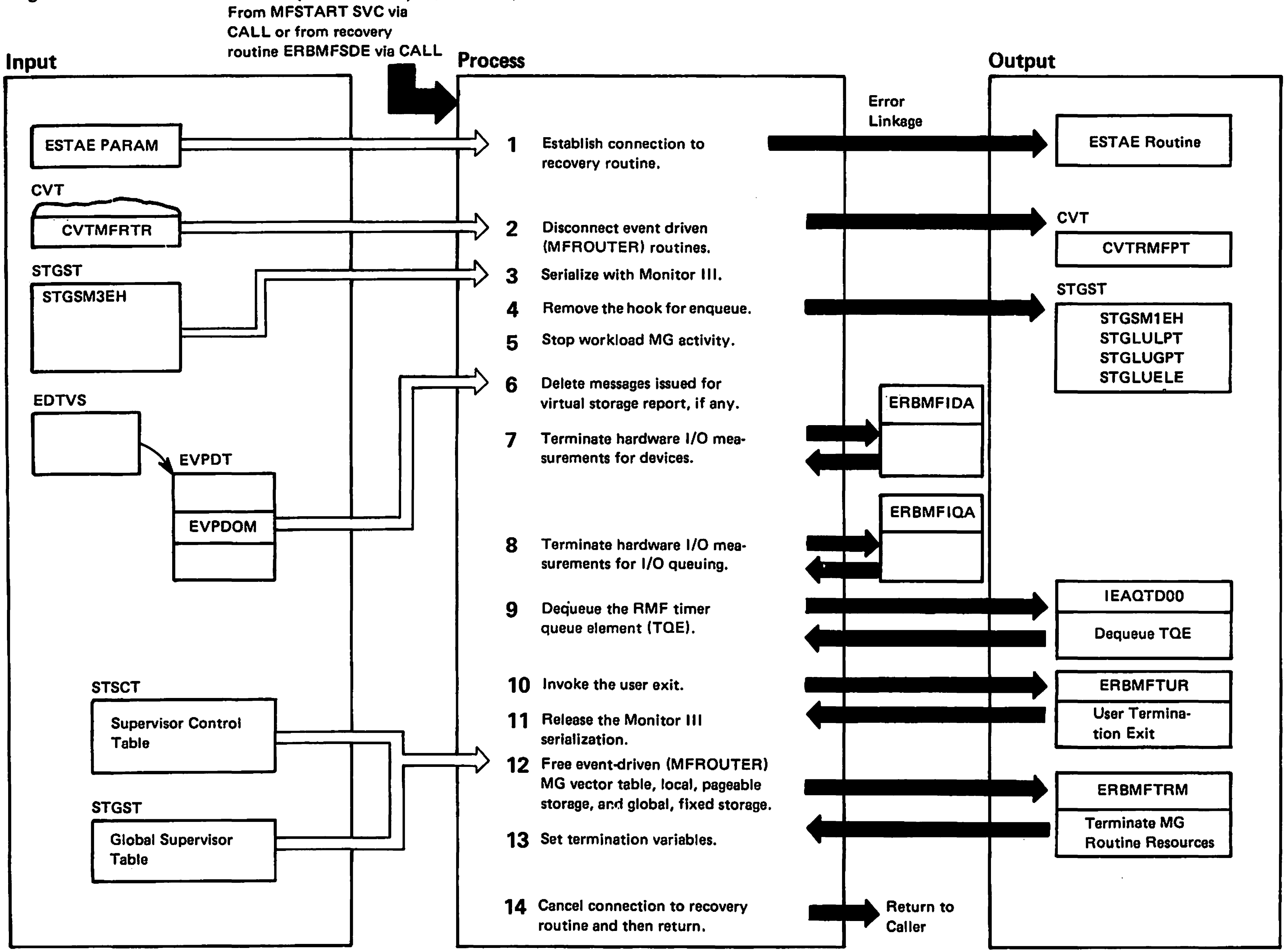
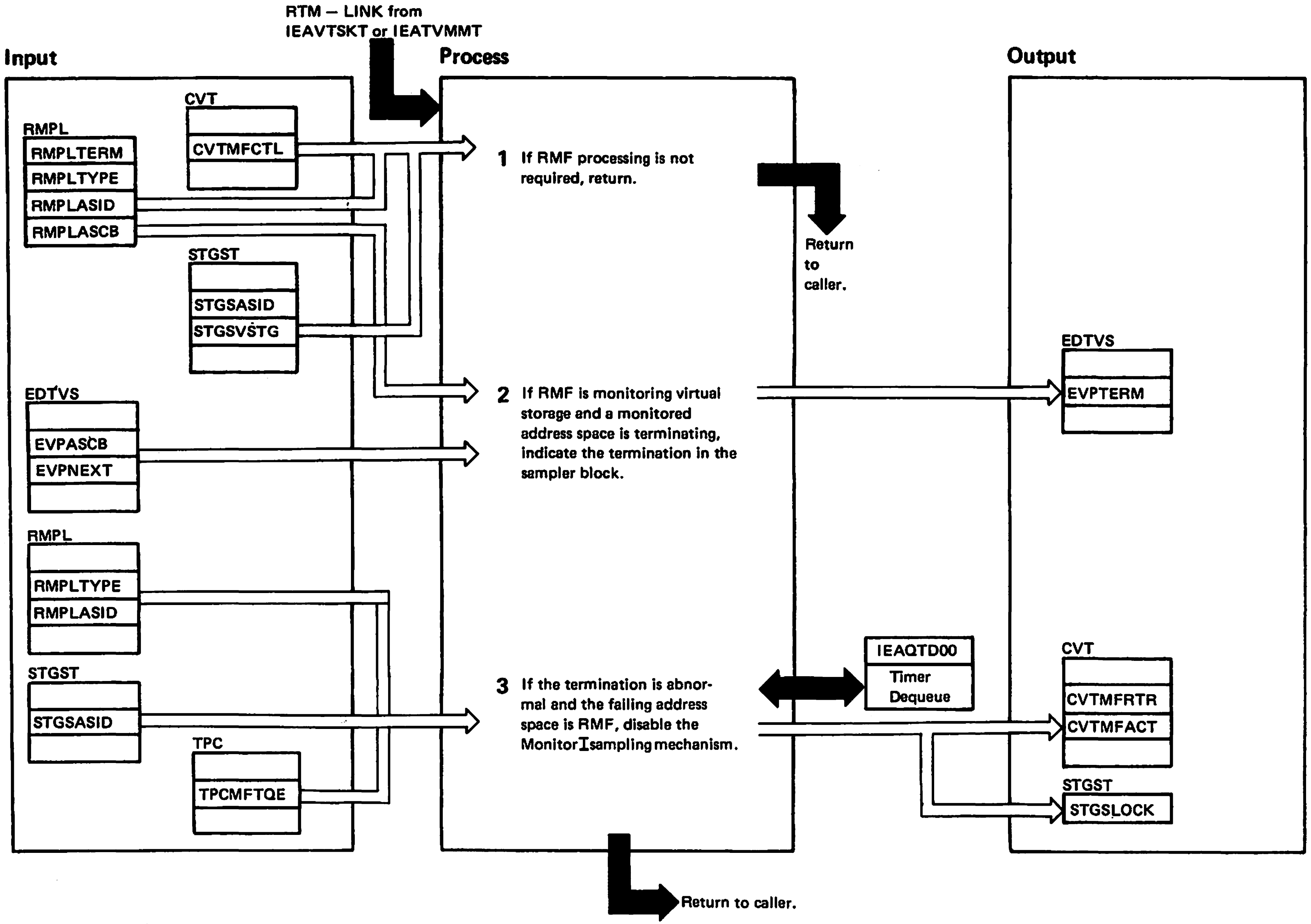


Diagram 16. Termination Processor (ERBMFTMA) (Part 2 of 2)

Extended Description	Module	Label
The Termination Processor (ERBMFTMA) disconnects RMF from the residence nucleus. The Termination Processor dequeues the Timer Queue Element (TQE), disconnects the event driven (cycle) MG routines, disables workload activity data collection, releases global storage, removes the address of the ENQUE sampler (ERBMFEEQ) from the CVT, and terminates hardware I/O measurements for non DASD or tape devices.	ERBMFTMA	
1 The Termination Processor provides ESTAE parameters to provide for retrying while releasing resources.	ERBMFTMA	
2 The linkage to the MFROUTER service routine (ERBMFEVT) is changed so that if an attempt is made to transfer control to ERBMFEVT, immediate return will be made by a BR 14. The Termination Processor also ensures that no CPU is currently executing event-driven MG code when this code is disconnected.	ERBMFTMA ERBMXTXR	
3 To serialize with Monitor III, the Termination Processor enqueues on the resource with a major name of RMF and a minor name of ENQ.NOTIFY.INTER.	ERBMFTMA	
4 The Termination Processor removes the hook for ENQUEUE. Thus, no subsequent SYSEVENT call causes a branch to the ENQUEUE event processing module (ERBMFEEQ). Therefore, ERBMFTMA clears the pointer to ERBMFEEQ (STGSM1EH) in the STGST. If Monitor III is not active (the STGSM3EH pointer is zero), clears the pointer to ERB3GLUE in the CVT (CVTRMFPT) and releases the working storage for ERB3GLUE.	ERBMFTMA	
5 The Termination Processor causes the workload manager to stop workload activity data collection.	ERBMFTMA IRARMWLM	
6 ERBMFDVP issues message ERB428E when a private area to be monitored is not active. If any messages are outstanding, delete them using DOM.	ERBMFDVP	
7 Call ERBMFIDA to terminate hardware I/O measurements for all non DASD or tape devices. Hardware measurements must be stopped before the event data control blocks (EDDDBs) are freed because the EDDDB contains the UCB pointer for the device. This pointer will be used to issue the MSCH that will turn off the MBI and MM bit settings.	ERBMFIDA	
8 Call ERBMFIQA to terminate hardware I/O measurements for I/O queuing.	ERBMFIQA	
9 The Termination Processor dequeues the RMF timer queue element (TQE) by disabling (using the SETLOCK macro instruction); providing a functional recovery routine (ERBMFFUR) link (because of having disabled); and using the TQE Dequeue routine (IEAQTD00) to dequeue the TQE. The Termination Processor then cancels the FRR link, and enables by means of the SETLOCK macro instruction.	ERBMFTMA IEAQTD00	
10 Invoke ERBMFTUR, the user termination exit.		
11 The Termination Processor frees the RMF resource used to serialize with Monitor III.	ERBMFTMA	
12 The Termination Processor calls routine ERBMFTRM to release the resources of each MG routine. The Termination Processor uses the FREEMAIN macro instruction to release the measurement Vector Table (STMMV), the RMF local storage, and RMF global storage.	ERBMFTRM ERBMFTMA	
13 The Termination Processor dequeues the RMF enqueue resource by use of the DEQ macro instruction.	ERBMFTMA	
14 The ESTAE connection is canceled by use of the ESTAE macro instruction.	ERBMFTMA	

Diagram 17. RMF Memory/Task Termination Resource Manager (ERBMFRES) (Part 1 of 2)



Extended Description	Module	Label

ERBMFRES is a resource manager installed by the user. The user puts the name of this module in CSECT IEAVTRML of load module IGC0001C. RTM invokes the modules listed in this CSECT for every address space or task termination. When required, ERBMFRES performs end-of-memory processing for RMF. Installation of ERBMFRES is described in the *RMF Program Directory* and in the *MVS/XA System Programming Library: System Modifications.*

1 ERBMFRES returns immediately to the caller when either task termination is requested or RMF was never active.

2 If the Monitor I virtual storage report is active and monitoring private areas, ERBMFRES compares the ASCB address of the terminating address space with the ASCB addresses in the private area sampler blocks (EDTVS). If there is a match, the module sets a bit in the sampler block to indicate the address space has terminated.

3 If termination is abnormal and the address space is RMF, (as would occur when, for example, RMF has been terminated by the FORCE command), ERBMFRES disables the Monitor I sampling process to prevent interference with any later attempt to restart RMF. The process of disabling the Monitor I sampler is as follows:

a) Resets CVTMFRTR and CVTMFACT.

b) Uses RISGNL to ensure that no other processors are executing RMF sampling routines.

c) Calls IEAQTD00 to dequeue the RMF TQE.

d) Clears STGSLOCK, the byte used to serialize the samples and the interval processing.

Recovery Processing

ESTAE routine RESESTAE covers steps 2 and 3 only. RESESTAE produces an SDUMP and then requests a retry so that processing will resume at the module exit or at the next resource to be handled.

Diagram 18. CPU Activity Initialization (ERBMFICP) (Part 1 of 6)

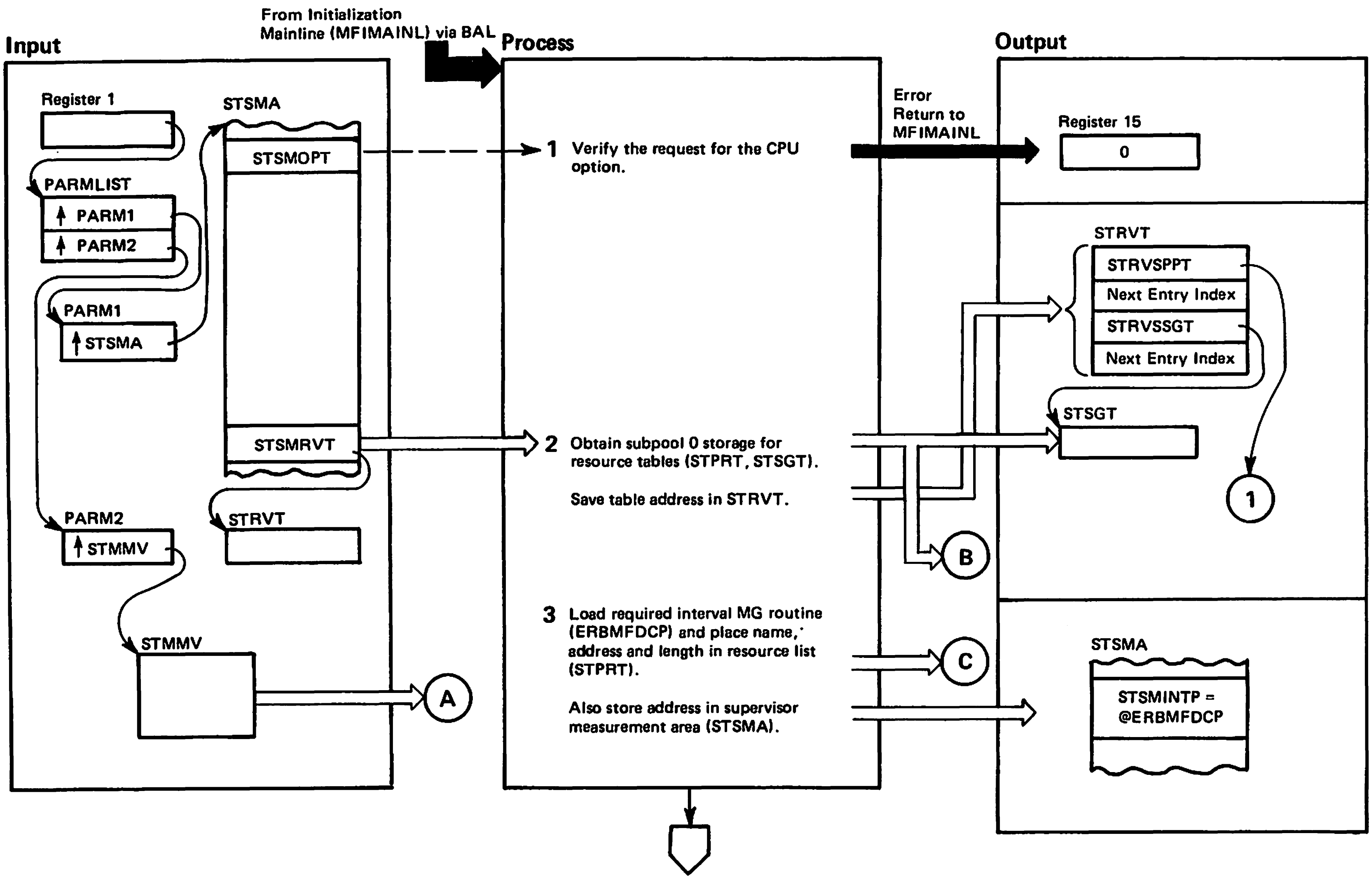


Diagram 18. CPU Activity Initialization (ERBMFICP) (Part 2 of 6)

Extended Description	Module	Label
The CPU Initialization (ERBMFICP) performs the initialization functions required to cause RMF to begin collecting CPU data. These functions include initializing both event-driven and interval-driven MG routines.		
1 The CPU initialization routine ensures that the input option for CPU has been specified by checking the STSMSTA bit in the STSMOPT word of the supervisor measurement area (STSMA). If the bit is on, the CPU option was requested, ERBMFICP continues with step 2. If the bit is off, ERBMFICP returns to the caller with a return code of 0 in register 15.	ERBMFICP	
2 ERBMFICP uses the GETMAIN macro instruction to obtain the necessary storage from subpool 0 for the program resource table (STPRT) and the storage resource table (STSGT). ERBMFICP saves their addresses in the STRVSPRT and STRVSSGT fields of the resource vector table (STRVT). It also initializes the STRVNPRT and STRVNSGT index fields in the STRVT. The index is used in subsequent processing to step through the contiguous entries in the STPRT and STSGT.	ERBMFICP	
3 ERBMFICP loads the interval MG routine (ERBMFDCP) into virtual storage space. An entry (name, address, and length) is added to the STPRT. Then the index for the next available STPRT entry is updated in the STRVT. The entry point address of ERBMFDCP is also placed in the STSMINTP field of the supervisor measurement area (STSMA). This address will be used by the MFDATA processor to give ERBMFDCP control.	ERBMFICP	

Diagram 18. CPU Activity Initialization (ERBMFICP) (Part 3 of 6)

Process

4 Load the event-driven MG routines (ERBMFEVT and ERBMFECP) into virtual storage. Place their names in the resource list (STPRT). Page fix the event-driven MG routines.

5 Save addresses of event-driven MG routines.

a. Store the MFROUTER service routine (ERBMFEVT) address into CVT.

Output

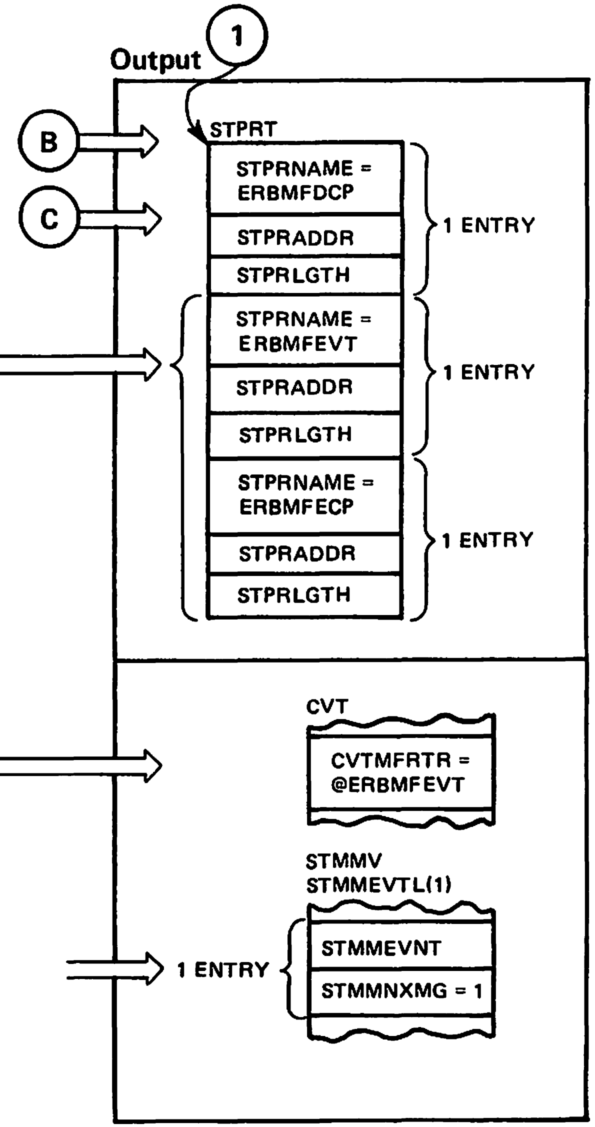


Diagram 18. CPU Activity Initialization (ERBMFICP) (Part 4 of 6)

Extended Description	Module	Label
4 ERBMFICP loads the MFROUTER processor (ERBMFEVT) and the event-driven MG module (ERBMFECP) into virtual storage. After their names, addresses and lengths (the length is made negative to indicate that a page free is needed should abnormal termination occur in any RMF function) are added to the STPRT, the index in the STRVT is updated to indicate the next available STPRT entry.	ERBMFICP	
ERBMFICP enters the page fix routine IEAVPSIB (address in CVT) to page fix ERBMFEVT and ERBMFECP. TCB=0 is specified to keep these modules fixed in LPA while the rest of RMF is swapped out.		
5 Save addresses of ERBMFEVT and ERBMFECP.	ERBMFICP	
a. The address of the MFROUTER service routine (ERBMFEVT) is stored into the CVTMFRTR field of the communications vector table (CVT). ERBMFEVT is entered in response to a timer interruption (every sample cycle).		

Diagram 18. CPU Activity Initialization (ERBMFICP) (Part 5 of 6)

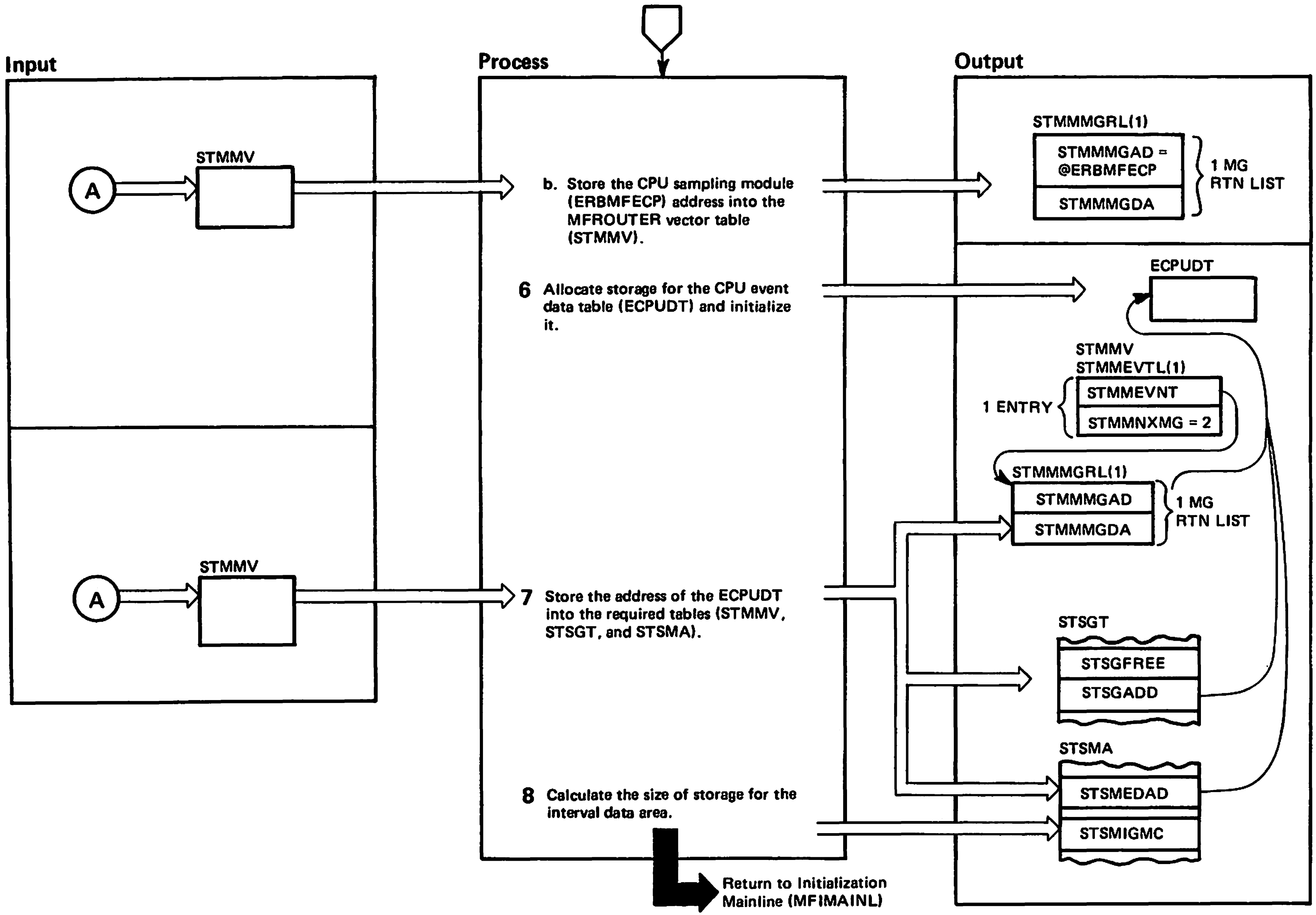
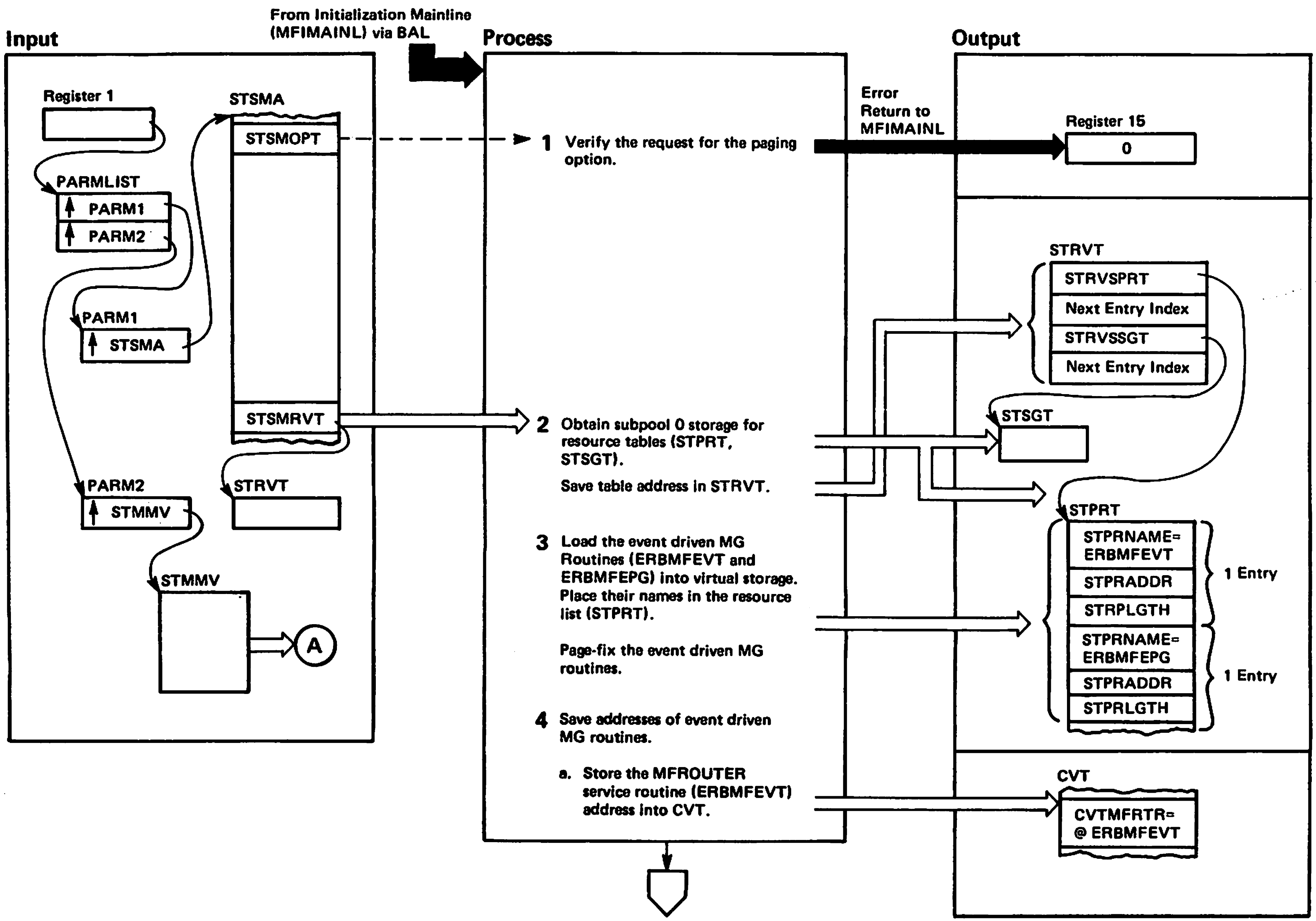


Diagram 18. CPU Activity Initialization (ERBMFICP) (Part 6 of 6)

Extended Description	Module	Label
b. The address of the CPU sampling module (ERBMFECP) is stored into the STMMMGRL (STMMMGAD) field of the STMMV. ERBMFECP is called by ERBMFEVT every sample cycle.		
6 ERBMFICP uses the GETMAIN macro instruction to obtain the global SQA fixed storage in SP245 for the CPU event data table (ECPUDT). This table contains address space analysis information which is gathered at each cycle. The table is zeroed and appropriate fields set to X'FF's (fields used for minimum values). The X'FF's indicate to ERBMFECP that no minimum values have been set yet.	ERBMFICP	
7 The address of the CPU event table (ECPUDT) is stored in:	ERBMFICP	
a. The MFROUTER measurement vector table (STMMV). The address of ECPUDT is saved in the STMMMGDA field of the STMMMGRL for use by the MFROUTER processor (ERBMFEVT). The index of the first event list (STMMNXMG field of STMMEVTL) is updated to the next STMMMGRL sampling MG routine slot.		
b. The storage resource table (STSGT). The address of ECPUDT is saved in the STSGADD field of the STSGT for later freeing by termination. The subpool and length of ECPUDT is saved in the STSGFREE field of the STSGT. The index to the next available STSGT entry is increased in the STRVT.		
c. The supervisor measurement area (STSMA). The address of the ECPUDT is saved in the STMEDAD field of the STSMA for use by the interval MG routine (ERBMFDCP).		

Extended Description	Module	Label
8 ERBMFICP calculates the length of the required interval data area. It stores the subpool in STSMISP and the length in STSMILEN of the STSMIGMC field of the STSMA. The storage length for CPU data is: 4 + length of (SMF70HDR) + length of (SMF70PRO) + length of (SMF70CTL) + ((CVTMAXMP+1) * length of (SMF70CPU) + length of (SMF70AID). This amount of storage provides room for a prefix control word and a CPU SMF record. ERBMFICP returns to the caller with a return code of 12 in register 15 to indicate sampling is in effect.	ERBMFICP	

Diagram 19. Paging Activity Initialization (ERBMFIPG) (Part 1 of 6)



1 Verify the request for the paging option.

2 Obtain subpool 0 storage for resource tables (STPRT, STSGT).

Save table address in STRVT.

3 Load the event driven MG Routines (ERBMFEVT and ERBMFEPG) into virtual storage. Place their names in the resource list (STPRT).

Page-fix the event driven MG routines.

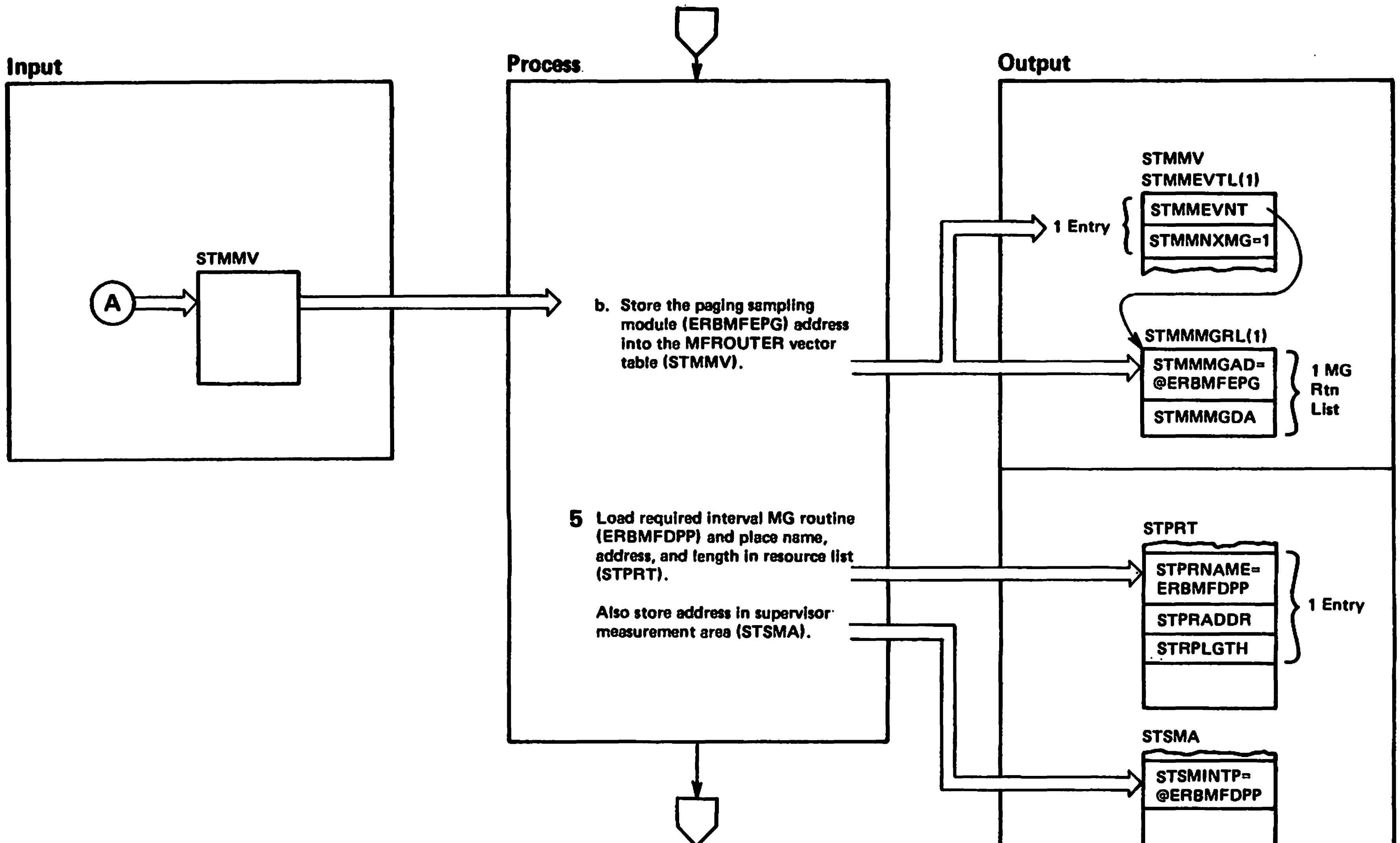
4 Save addresses of event driven MG routines.

a. Store the MFROUTER service routine (ERBMFEVT) address into CVT.

Diagram 19. Paging Activity Initialization (ERBMFIPG) (Part 2 of 6)

Extended Description	Module	Label
Paging Initialization (ERBMFIPG) performs the initialization functions required to cause RMF to begin collecting paging data. These functions include initializing both event-driven and interval-driven MG routines.		
1 ERBMFIPG ensures that the input option for paging has been specified by checking the STSMSTA bit in the STSMOPT word of the supervisor measurement area (STSMA). If the bit is on, the paging option was requested, ERBMFIPG continues with Step 2. If the bit is off, ERBMFIPG returns to the caller with a return code of 0 in register 15.	ERBMFIPG	
2 ERBMFIPG uses the GETMAIN macro instruction to obtain the necessary storage from SP0 for the program resource table (STPRT) and the storage resource table (STSGT). ERBMFIPG addresses in the STRVSPRT and STRVSSGT fields of the resource vector table (STRVT). It also initializes the STRVNPRT and STRNVSGT index fields in the STRVT. The index is used in subsequent processing to step through the contiguous entries in the STPRT and STSGT.	ERBMFIPG	
3 ERBMFIPG loads the MFROUTER processor (ERBMFEVT) and the event-driven MG module (ERBMFEPG) into virtual storage. After their names, addresses and lengths (the length is made negative to indicate that a page free is needed should abnormal termination occur in any RMF function) are added to the STPRT, the index in the STRVT is updated to indicate the next available STPRT entry.	ERBMFIPG	
ERBMFIPG enters the page fix routine (PGSER) via a branch to IEAVPSIB (address in CVT) to page fix ERBMFEVT and ERBMFEPG. TCB=0 is specified to keep these modules fixed in LPA while the rest of RMF is swapped out.		
4 Saves addresses of ERBMFEVT and ERBMFEPG.	ERBMFIPG	
a. The address of the MFROUTER service routine (ERBMFEVT) is stored into the CVTMFRTR field of the communications vector table (CVT). ERBMFEVT is entered in response to a timer interruption (every sample cycle).		

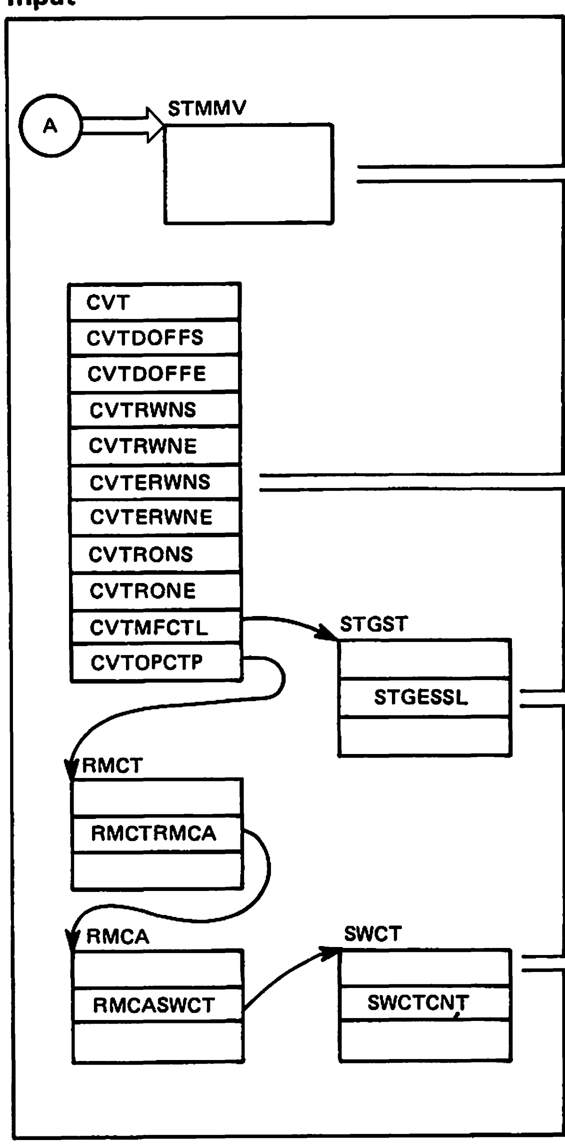
Diagram 19. Paging Activity Initialization (ERBMFIPG) (Part 3 of 6)



Extended Description	Module	Label
4 (continued)		
b. The address of the paging sampling module (ERBMFEPG) is stored into the STMMMGRL (STMMMGAD) field of the STMMV. ERBMFEPG is called by ERBMFEVT every sample cycle.		
5 ERBMFIPG loads the interval MG routine (ERBMFDPP) into virtual storage space. An entry (name, address, and length) is added to the STPRT. Then the index for the next available STPRT entry is updated in the STRVT. The entry point address of ERBMFDPP is also placed in the STSMINTP field of the supervisor measurement area (STSMA).	ERBMFIPG	

Diagram 19. Paging Activity Initialization (ERBMFIPG) (Part 5 of 6)

Input



Process

6 Allocate storage for the paging event data table (EPPEDT) and initialize it.

7 Store the address of the EPPEDT into the required tables (STMMV, STSGT, and STSMA).

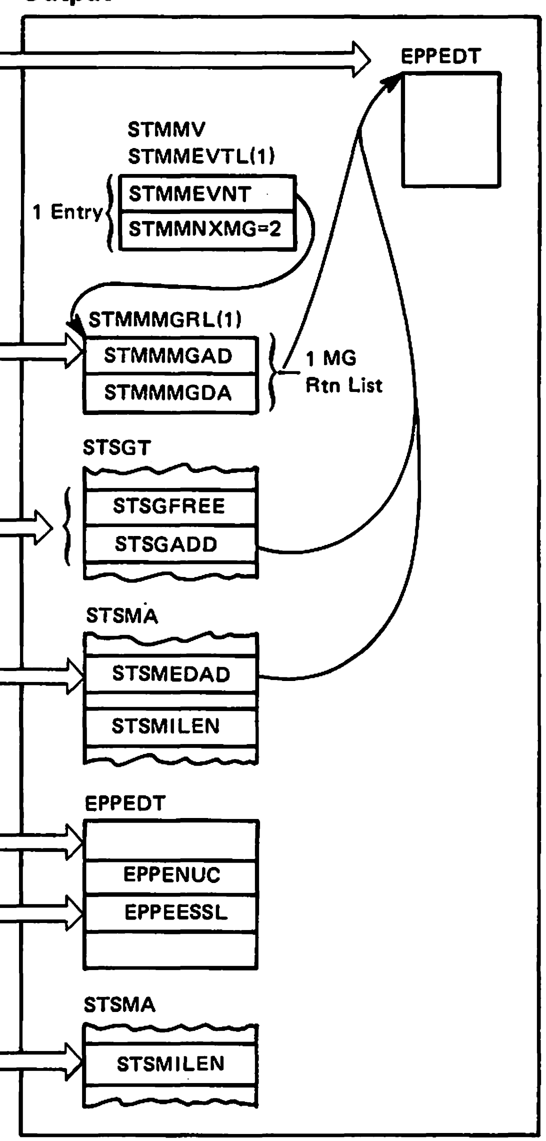
8 a. Calculate the number of real storage frames occupied by the nucleus.

b. Obtain the number of extended storage frames installed.

9 Calculate the size of storage for the interval data area.

Return to Initialization Mainline (MFIMAINL)

Output



Extended Description	Module	Label
6 ERBMFIPG uses the GETMAIN macro instruction to obtain the global SQA fixed storage in SP245 for the paging event data table (EPPEDT). This table contains paging information which is gathered at each cycle. The table is zeroed and appropriate fields set to X'FF's (fields used for minimum values).	ERBMFIPG	
7 Stores the address of the paging event table (EPPEDT) in: a. The MFROUTER measurement vector table (STMMV). The addresss of EPPEDT is saved in the STMMMGDA field of the STMMMGRL for use by the MFROUTER processor (ERBMFEVT). The index of the first event list (STMMNXMG field of STMMEVTL) is updated to the next STMMMGRL sampling MG routine slot. b. The storage resource table (STSGT). The address of EPPEDT is saved in the STSGADD field of the STSGT for later freeing by termination. The subpool and length of EPPEDT is saved in the STSGFREE field of the STSGT. The index to the next available STSGT entry is incremented in the STRVT. c. The supervisor measurement area (STSMA). The address of the EPPEDT is saved in the STMEDAD field of the STSMA for use by the interval MG routine (ERBMFDPP).	ERBMFIPG	

Extended Description	Module	Label
8 a. Calculates the number of nucleus frames as follows: (DAT-off nucleus end address – DAT-off nucleus store address)/page size + (Read/Write RMODE=24 nucleus end address – Read/Write RMODE=24 nucleus start address)/ page size + (Read/Write RMODE=ANY nucleus end address – Read/Write RMODE=ANY nucleus start address)/ page size + (Read only nucleus end address – Read only nucleus start address)/page size. b. Moves the number of installed extended storage frames from the global supervisor table (STGST), where ERBMFMFC has stored it, into the EPPEDT, for use by the paging gatherer.		
9 ERBMFIPG calculates the length of the required interval data area, which must be large enough to hold a prefix control word and a paging SMF record. The calculation used to determine the length is: 4 + length of (SMF71HDR) + length of (SMF71PRO) + length of (SMF71PAG) + n * length of (SMF71SWP) where n represents the number of SRM swap reasons (obtained from the SRM swap control table, SWCT). ERBMFIPG then returns to the caller with a return code of 12 in register 15 to indicate that sampling is in effect.		

Diagram 20. Workload Activity Initialization (ERBMFIWK) (Part 1 of 8)

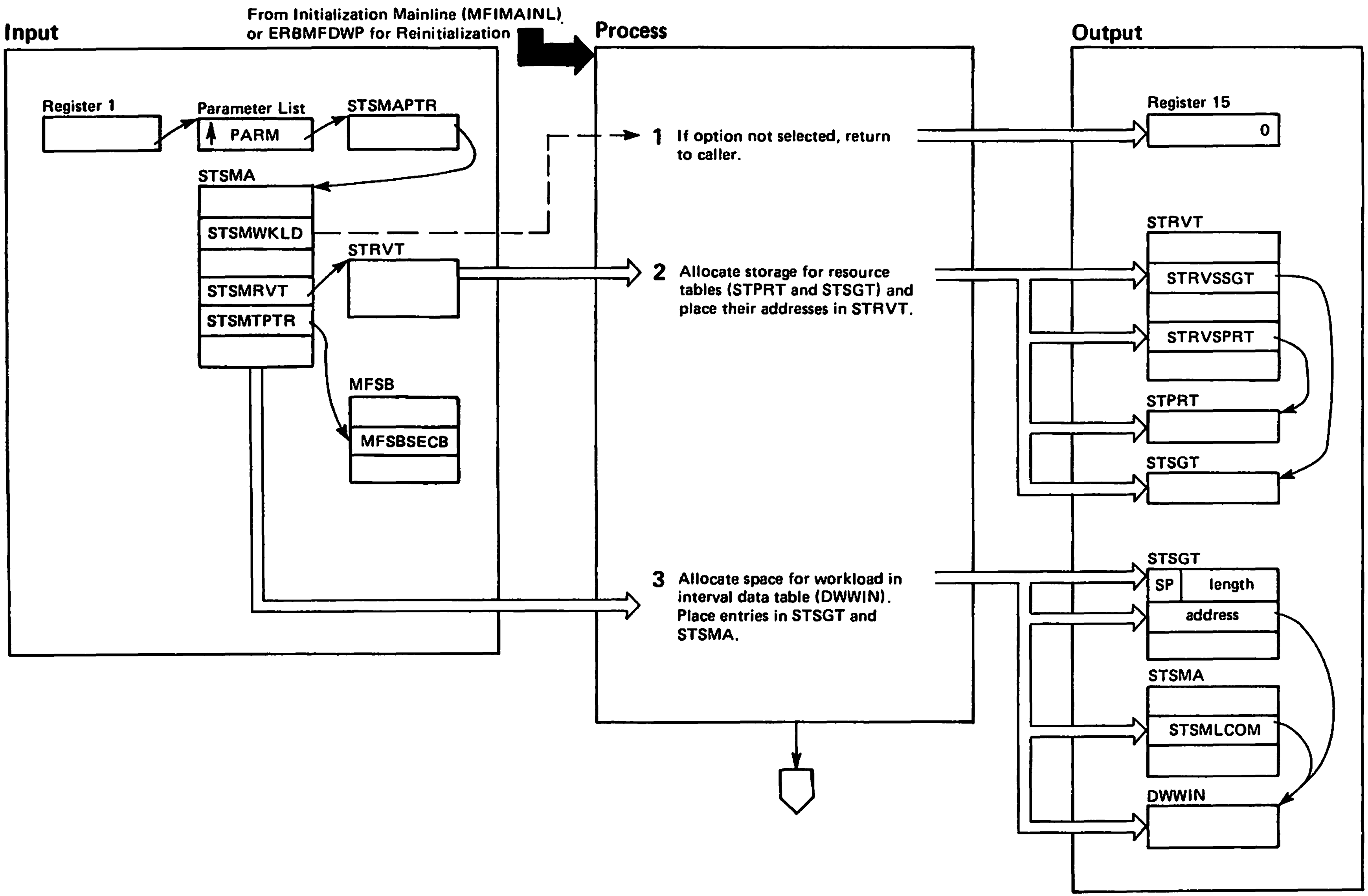


Diagram 20. Workload Activity Initialization (ERBMFIWK) (Part 2 of 8)

Extended Description	Module	Label
The workload initialization routine (ERBMFIWK) allocates storage for control blocks, ensures that copies of the interval measurement gathering routine (ERBMFDWP) and the workload activity routine (IRARMWAR) are in storage, obtains storage for the workload activity measurement tables (WAMT and ICSM) and initiates the collection of workload activity data.		
1 ERBMFIWK tests to ensure that the workload option has been selected by inspecting the STSMSTA field in the supervisor measurement area (STSMA).	ERBMFIWK	
2,3 ERBMFIWK uses the GETMAIN macro to obtain the storage required for the resource tables (STPRT and STSGT) and for the workload interval table (DWWIN).	ERBMFIWK	

Diagram 20. Workload Activity Initialization (ERBMFIWK) (Part 3 of 8)

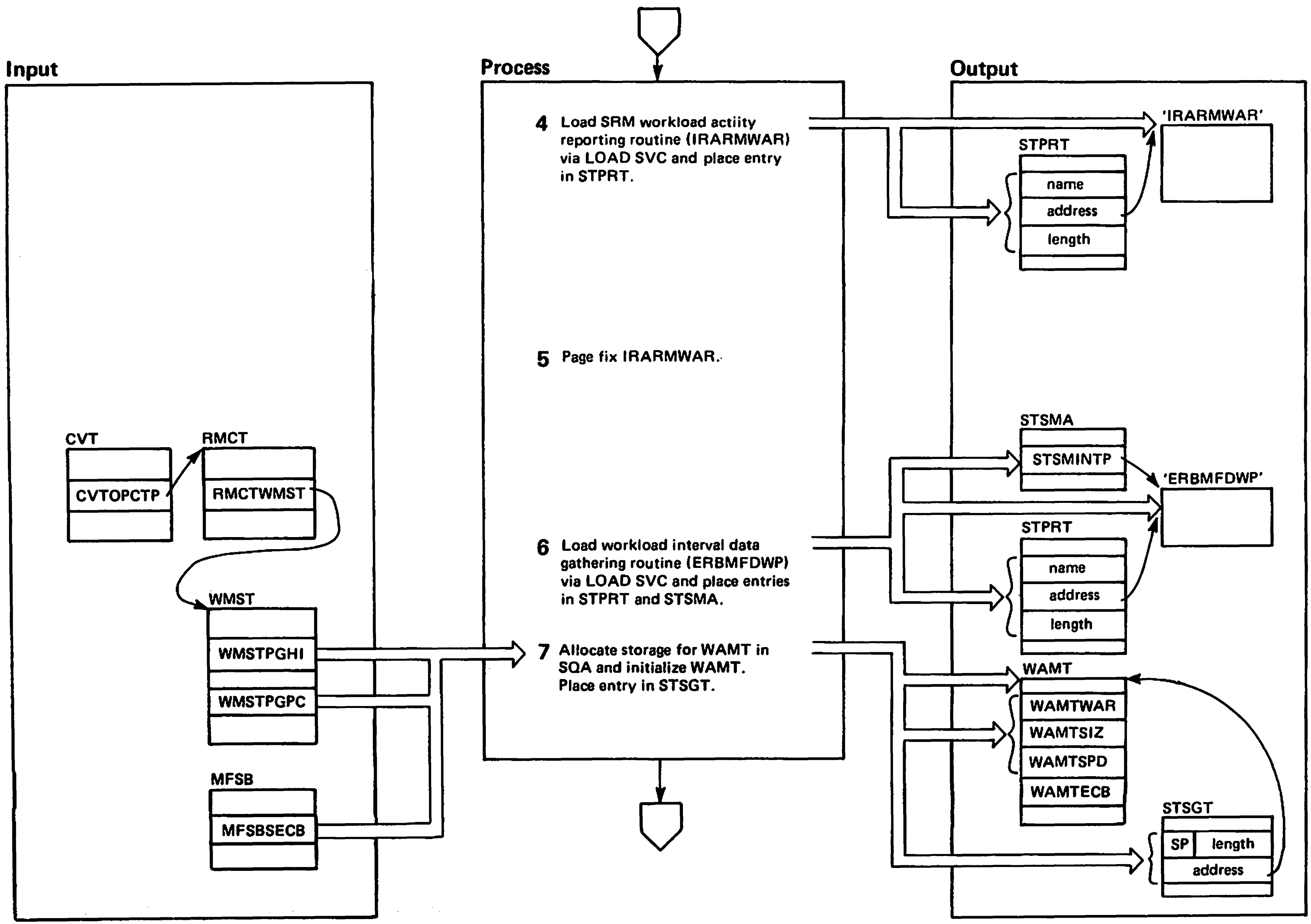


Diagram 20. Workload Activity Initialization (ERBMFIWK) (Part 4 of 8)

Extended Description	Module	Label
4 ERBMFIWK uses the LOAD macro to obtain a copy of the workload activity routine IRARMWAR. The entry address of IRARMWAR as well as its name and length are placed in the program resource table (STPRT).	ERBMFIWK	
5 ERBMFIWK page fixes module IRARMWAR (IRARMWAR runs in a disabled state and cannot take a page fault) via a branch entry to the page fix routine (PGSER).	ERBMFIWK	
6 ERBMFIWK uses the LOAD macro to obtain a copy of the workload activity measurement gathering routine (ERBMFDWP). It places the entry address of ERBMFDWP in the STSMINTP field of the STSMA and places an entry in the program resource table (STPRT) for eventual use by ERBMFDWP.	ERBMFIWK	
7 ERBMFIWK calls subroutine MFIIPSWA to obtain storage for the workload activity measurements table (WAMT). MFIIPSWA calculates the size of the WAMT as follows: (length of WAMT header) + (number of highest performance group) * (length of WAMT index entry) + (number of defined performance group periods) * (length of WAMT entry for a performance group). It then obtains space for the WAMT by issuing an unconditional GETMAIN, accounts for this storage in the resource table (STSGT) and initializes the WAMTWAR, WAMTSIZ, WAMTSPD, and WAMTECB fields of the WAMT for use by IRARMWAR.	ERBMFIWK	MFIIPSWA

Diagram 20. Workload Activity Initialization (ERBMFIWK) (Part 5 of 8)

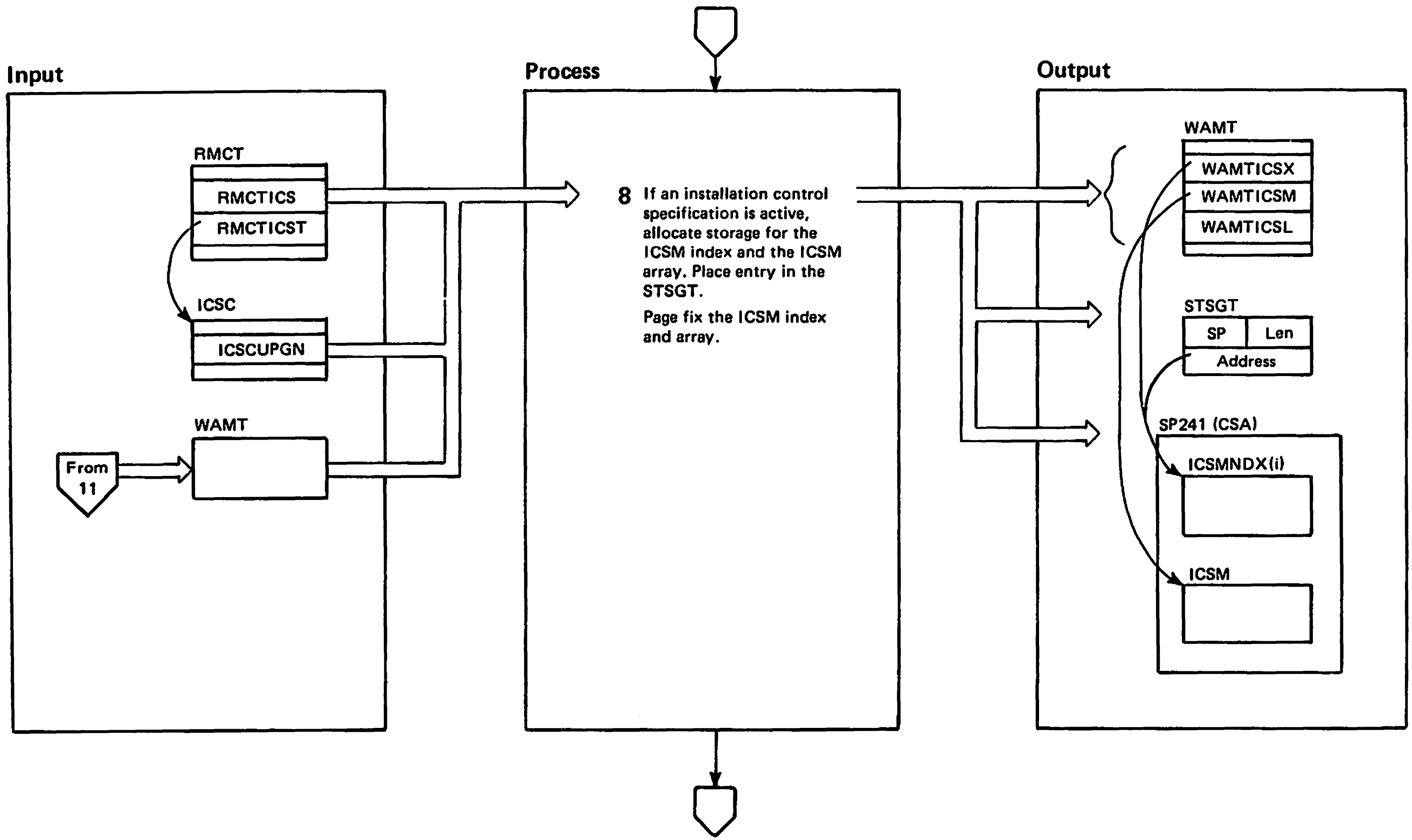


Diagram 20. Workload Activity Initialization (ERBMFIWK) (Part 6 of 8)

Extended Description	Module	Label
8 If an installation control specification is active (RMCTICS = ON), ERBMFIWK obtains storage for the ICSM index (ICSMNDX) and the ICSM array from subpool 241. The size of the area is calculated as follows: (highest PGN+1) * length (ICSMNDX) + (length of one ICSM element) * the number of unique PGNs (from ICSCUPGN) ERBMFIWK then accounts for this storage in the resource table (STSGT) and initializes the following fields in the WAMT for use by IRARMWAR and ERBMFDWP (RMF interval driven module): • WAMTICSX = address of ICSMNDX • WAMTICSM = address of ICSM • WAMTICSL = length of ICSMNDX Issue the PGSER macro to page-fix the ICSM index and array. This is necessary because the SYSEVENT routine, invoked in the next step to initiate workload data collection, runs in the disabled state.	ERBMFIWK	

Diagram 20. Workload Activity Initialization (ERBMFIWK) (Part 7 of 8)

Input

WAMT

ICSMNDX (i)

ICSM

Process

9 Issue SYSEVENT to initiate SRM workload data collection. Set the flag in the workload interval data table (DWWIN) to indicate that the IPS started in the interval.

10 Page-free the ICSM index and array.

11 If unsuccessful initiation (RC ≠ 0), issue an ABEND macro with code '36D'X. Reason code in Reg 15 is return code from SYSEVENT.

12 Save length of WAMT in workload interval data table (DWWIN).

13 Return to caller.

Return to Caller (MFIMAINL or ERBMFDWP)

Output

DWWIN

DWWINIPS

Register 15

Reason Code

DWWIN

DWWIWAML

Register 15

0

Diagram 20. Workload Activity Initialization (ERBMFIWK) (Part 8 of 8)

Extended Description	Module	Label
9 Subroutine MFIIPSWA attempts to initiate workload activity data collection by invoking the SRM via SYSEVENT. It sets the DWWINIPS field in the workload interval table (DWWIN) to indicate that the IPS started in the interval and then returns to the caller. A return code of zero from MFIIPSWA indicates successful initialization. A non-zero return code signifies unsuccessful initialization.	ERBMFIWK	MFIIPSWA
10 Issue the PGSER macro to page-free the ICSM index and array for paging.		
11 If the return code from MFIIPSW is anything other than zero, ERBMFIWK issues an ABEND with a code of '36D'X and a reason code which is the return code from SYSEVENT. Otherwise step 16 executes next.	ERBMFIWK	
12 If return from MFIIPSW is zero, indicating success, then the length of the WAMT is placed in the DWWIWAML field of the workload interval table (DWWIN) for later use by ERBMFDWP.		
13 Return.		

Error Processing:
None.

Diagram 21. Channel Path Initialization (ERBMFIHA) (Part 1 of 4)

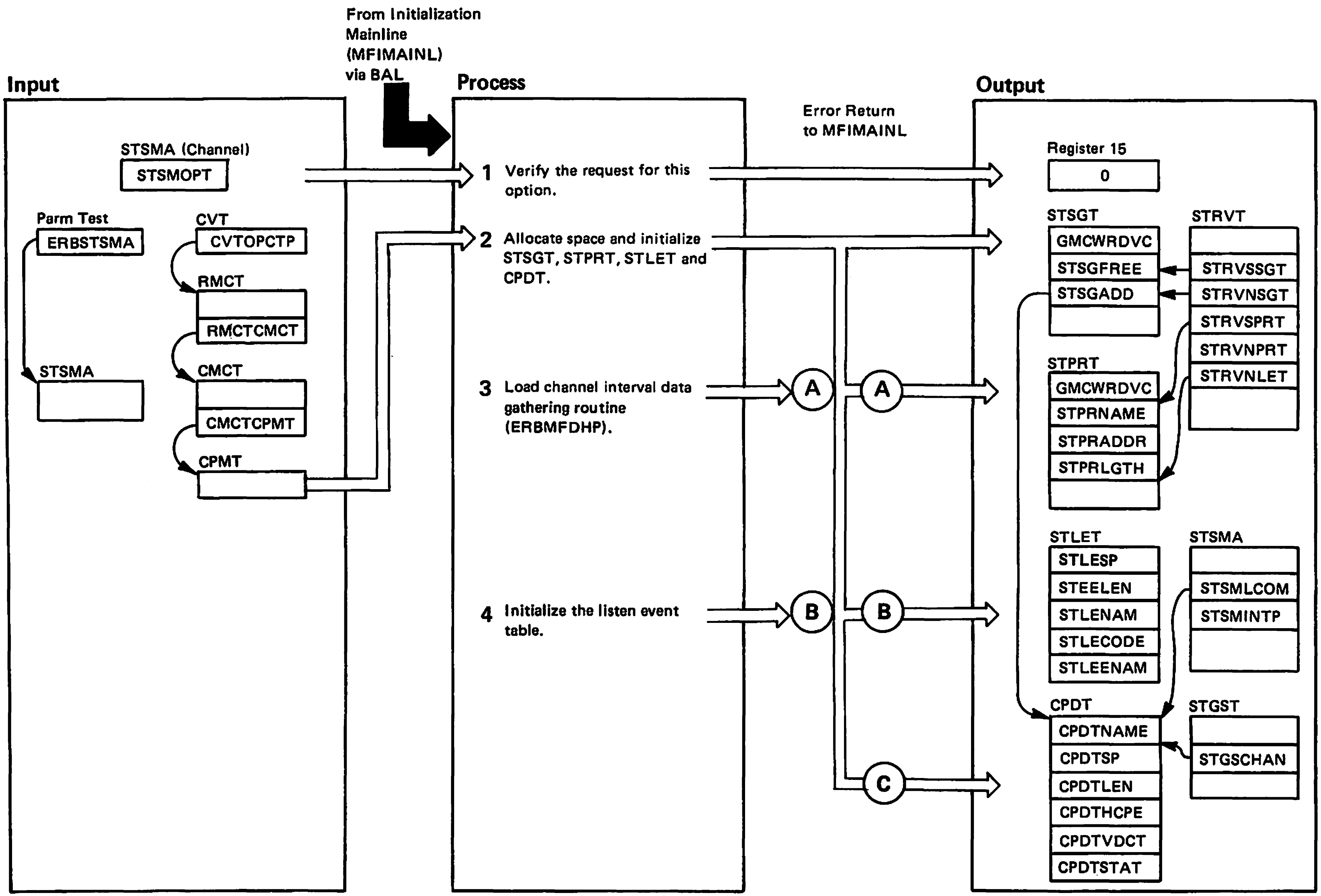


Diagram 21. Channel Path Initialization (ERBMFIHA) (Part 2 of 4)

Extended Description	Module	Label

The Channel Initialization routine (ERBMFIHA) allocates storage for control blocks, ensures that a copy of the interval measurement gathering (STSGT) routine (ERBMFDHP) is in storage, requests activation of the listening exit table and fills control blocks with initial values.

1 ERBMFIHA tests to insure that the channel option has been specified by inspecting the STSMOPT field in the supervisor measurement area (STSMA). If the bit is on channel measurement is requested, ERBMFIHA continues with Step 2. If the bit is off ERBMFIHA returns to the caller with a return code of 0 in register 15.

2 ERBMFIHA builds the control blocks, STPRT and STSGT, used to store resource information and the listening event table, STLET. The addresses of these areas are stored in the resource vector table (STRVT).

The channel path data table (CPDT) is also created and its addresses stored in the STGSCHAN field of the global supervisor table (STGST). The CPDTs address, subpool number, and length are saved in the storage resource table (STSGT).

The length of the CPDT is calculated using the highest value CPID number, obtained from the channel path measurement table (CPMT), and multiplying it by the length of a CPDT entry.

3 Load the channel interval data gathering routine (ERBMFDHP).

The name, address, and length of this routine is stored in the STPRT, and the STRVNPRT is updated to show the addition of ERBMFDHP. The address of ERBMFDHP is also stored into STSMINTP of the STSMA for use by MFDATA (IGX00022).

4 Initialize the listening event table (STLET) with the parameters necessary to activate the listening exit:

- The event code for channel path status is 9.
- The listening exit routine name is ERBLXCHP.

Diagram 21. Channel Path Initialization (ERBMFIHA) (Part 3 of 4)

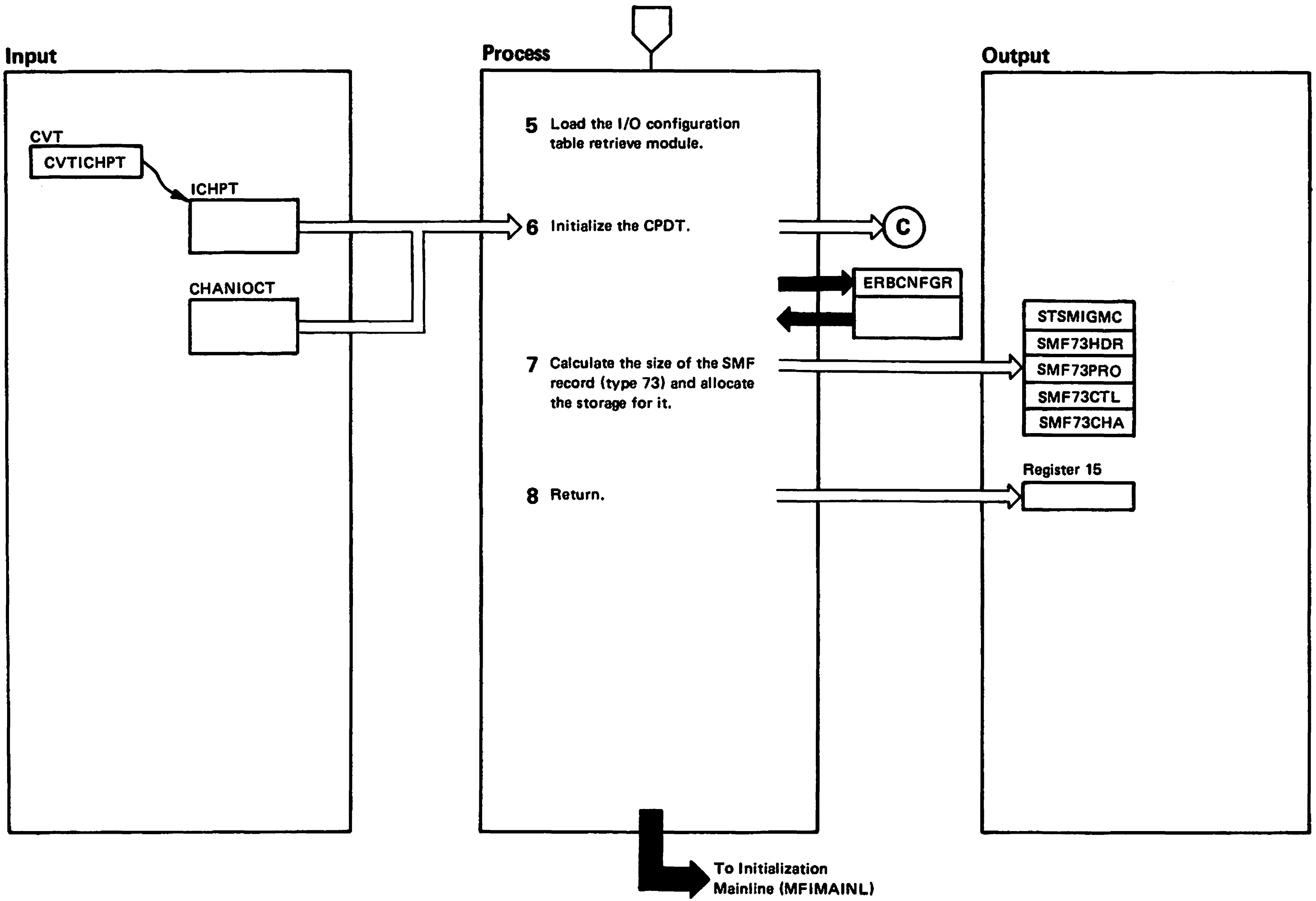


Diagram 21. Channel Path Initialization (ERBMFIHA) (Part 4 of 4)

Extended Description	Module	Label

5 Load the I/O configuration table retrieve module (ERBCNFGR) and save its address in CNFGRPTR.

6 Initialize the CPDT with values from the ICHPT and the data returned by ERBCNFGR.

ERBMFIHA loops through all the entries of the ICHPT to determine which channel path IDs are installed in the system. The information in the ICHPT is used to set the valid and on line flags in the CPDT entry. A count of all the valid CPIDs is kept in CPDTVDCT. This number determines the number of SMF record data sections to reserve.

For each CPID, ERBMFIHA calls ERBCNFGR to obtain the channel type.

The channel type returned will be used to set the CPDTBY and CPDTBL flags in the CPDT.

7 Calculate the size of the SMF record (type 73) and allocate the storage for it.

The size of the area occupied by the SMF record is calculated as follows:

(length of STSMIGMC) + (length of SMF73HDR) + (length of SMF73PRO) * (number of product sections) + (length of SMF73CTL) * (number of control sections) + (length of SMF73CHA) * (number of data sections)

Each SMF record has

- one header section
- one product section
- one control section
- as many data sections as there are installed CPIDs

8 Return to the caller with one of the following codes:

X'00' No channel report is requested
X'20' A channel report is requested. This return code signals IGX00022 that activation of a listening exit for channel path is requested.

Diagram 22. Device Initialization (ERBMFIDV) (Part 1 of 12)

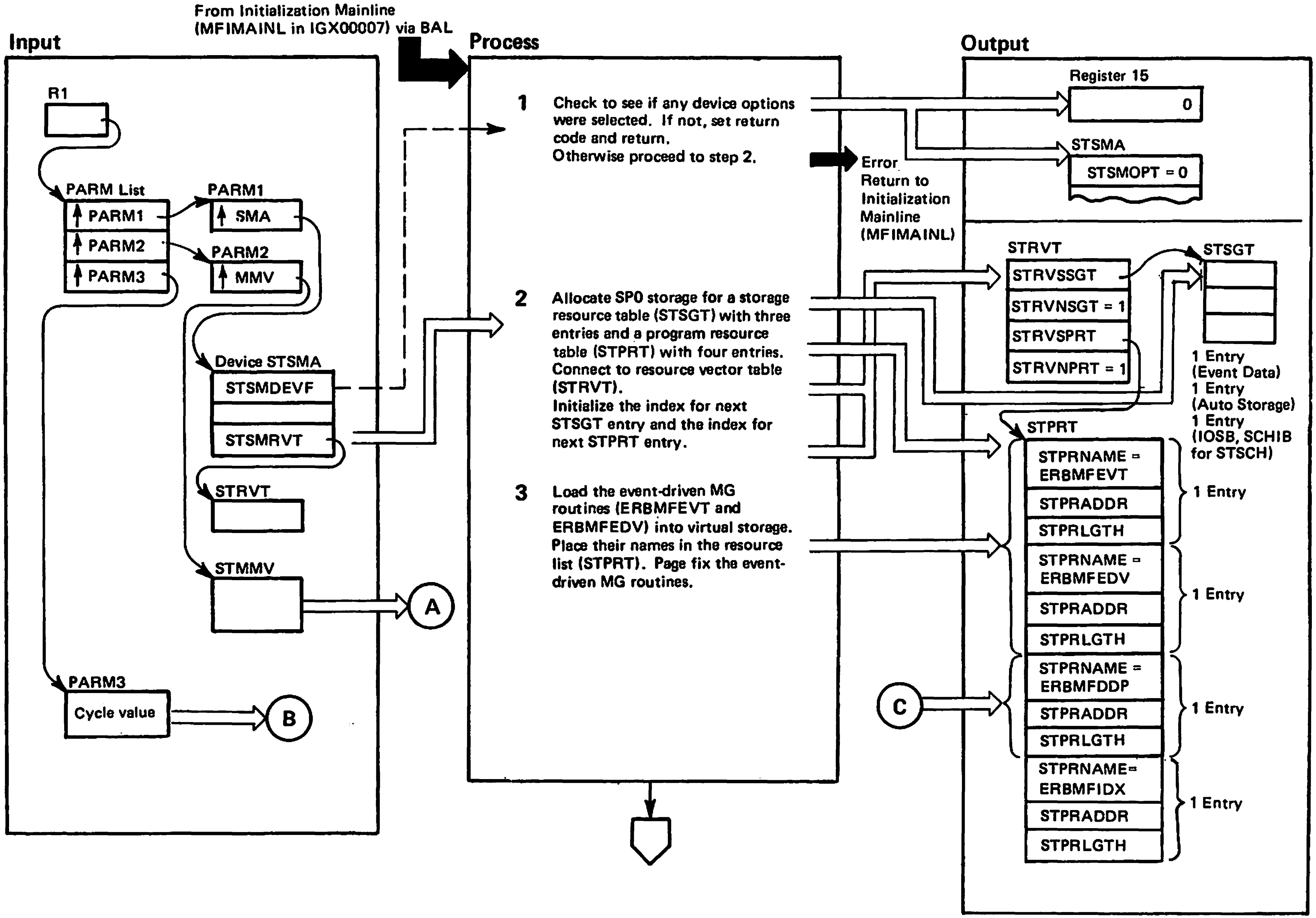


Diagram 23. Trace Activity Initialization (ERBMFITR) (Part 4 of 4)

Extended Description	Module	Label
7 A storage resource table entry is filled in for the system queue area storage and a pointer to it placed in STSMEDAD.	ERBMFITR	
8 The system queue area storage is now divided into trace entries, one for each variable traced. The algorithm used in step 6 is used to determine the boundaries and chain the entries together. The address of each field in storage is determined from the control block structure from SRM, ASM, and RSM (declared as part of the ERBMFITR module). The interval data length is determined as follows: (2*area required for interval data) + (size of all MFTRTRAC control blocks) + (size of all SMF76HDR, SMF76PRO, SMF76A, and SMF76B parts) + (size of the work area for ERBMFETR). The sum is inserted into STSMILEN in the STSMA for ERBMFDTP.	ERBMFITR	
9 Invoke the user exit so it can provide an address for any user names provided.	ERBTRACE	
10 A return code of 12 is set to indicate that sampling is required and that MFROUTER (ERBMFEVT) is started. If trace is not initialized, a return code of 0 is set.	ERBMFITR	

Diagram 24. Page/Swap Dataset Activity Initialization (ERBMFISP) (Part 1 of 6)

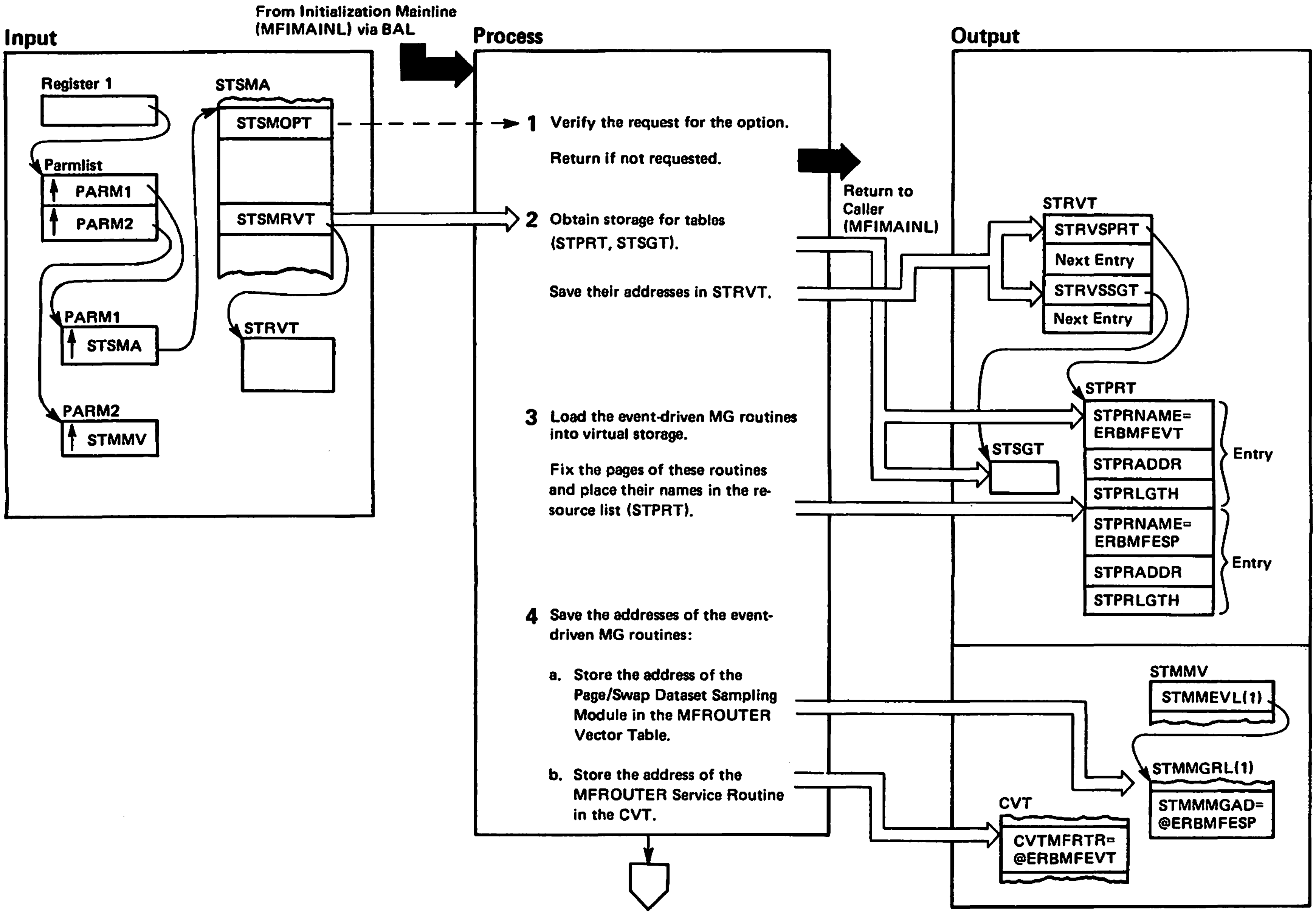
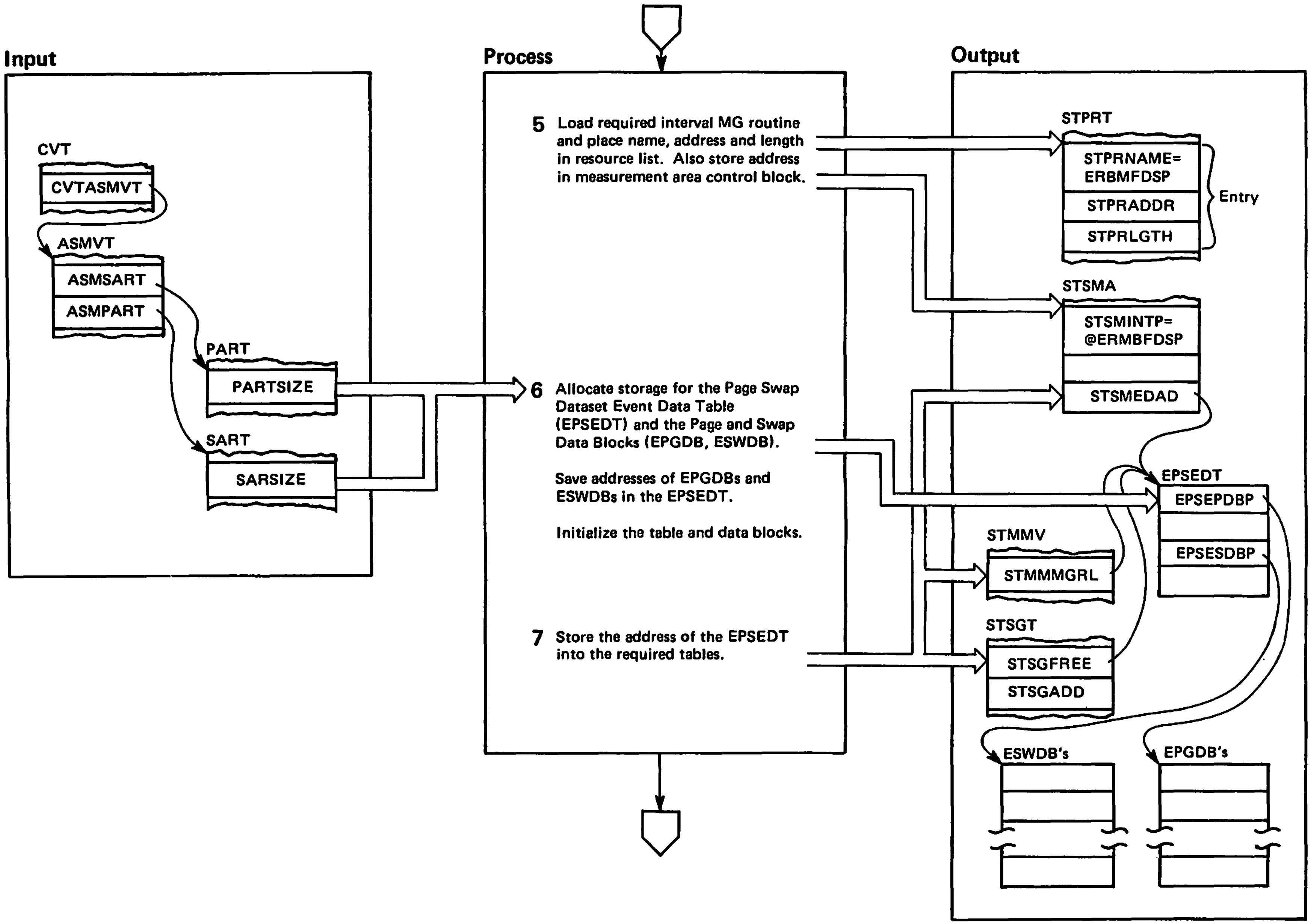


Diagram 24. Page/Swap Dataset Activity Initialization (ERBMFISP) (Part 2 of 6)

Extended Description	Module	Label
The Page/Swap Dataset Initialization (ERBMFISP) performs the initialization allowing RMF to begin collecting page/swap dataset information. Initialization for both event-driven and interval-driven MG routines is accomplished.		
1 ERBMFISP ensures that the input option has been specified by checking the STSMSTA bit in the STSMOPT word of the Supervisor Measurement Area (STSMA).	ERBMFISP	
2 ERBMFISP uses the GETMAIN macro instruction to obtain the necessary storage from SP0 for the program resource table (STPRT) and the storage resource table (STSGT). It then saves the table addresses in the STRVSPRT and STRVSSGT fields of the resource vector table (STRVT). It also sets an index of 1 in the STRVNPRT and STRVNSGT fields of the STRVT. This index is used to process the contiguous entries within the STPRT and STSGT.	ERBMFISP	
3 ERBMFISP loads the MFROUTER Processor (ERBMFEVT) and the event-driven MG routine (ERBMFESP) into virtual storage. After their entries are added to the STPRT, the index in the STRVT is updated to indicate the next available STPRT entry.	ERBMFISP	
ERBMFISP enters the page fix routine PGSER (address in CVT) to page fix ERBMFEVT and ERBMFESP, which run disabled and cannot tolerate page faults. TCB=0 is specified for the page fix to keep these modules fixed in LPA while the rest of RMF is available for swap out.		

Extended Description	Module	Label
4 The address of the Page/Swap Data Set Sampling Module (ERBMFESP) is stored into the STMMMGRL (STMMMGAD) field of the MFROUTER measurement vector table (STMMV).	ERBMFISP	
The address of the MFROUTER service routine (ERBMFEVT) is stored in the CVTMFRTR field of the communications vector table (CVT). ERBMFEVT is entered in response to a timer interruption (every sample cycle).		

Diagram 24. Page/Swap Dataset Activity Initialization (ERBMFISP) (Part 3 of 6)



Extended Description	Module	Label
5 ERBMFISP loads the interval MG routine (ERBMFDSP) into virtual storage space. An entry is added to the STPRT. Then the next available entry index is updated in the STRVT. The entry point address of ERBMFDSP is also placed in the STSMA (specifically STSMINTP). IGX00022 will use this address to call ERBMFDSP at the end of an interval.	ERBMFISP	
6 ERBMFISP uses the GETMAIN macro instruction to obtain the global SQA fixed storage in SP245 for the page/swap dataset event table (EPSEDT) and the page and swap dataset blocks (EPGDB's and ESWDB's). The data blocks will hold page/swap dataset information gathered at each cycle.	ERBMFISP	
The page/swap dataset initialization routine uses the GETMAIN macro instruction to obtain the global SQA. Enough space is reserved to contain a data block for each page and each swap dataset which could possibly be allocated for the current IPL. Counts of the maximum possible page and swap datasets are found in the paging activity reference table (PART – for page datasets) and the swap activity reference table (SART – for swap datasets). The fields PARTSIZE and SARSIZE reflect the number of PART and SART entries allocated at NIP time. The EPGDB and ESWDB data blocks form 2 queues – one for page datasets and one for swap datasets.	ERBMFISP	
The EPSEDT is zeroed and made to point to the first contiguous EPGDB and the first contiguous ESWDB. The data blocks are zeroed and the appropriate fields initialized to X'FF's (fields used for minimum values – see ERBEPSED mapping).		

Extended Description	Module	Label
7 The address of the page/swap dataset event table (EPSEDT) is stored into the STMMMGRL (STMMMGDA) of the MFROUTER measurement vector table (STMMV) for use by the MFROUTER Processor (ERBMFEVT) to invoke samplers. It then updates the index of the first event list (STMMNXT field of STMMEVTL) to the next available STMMMGRL sampling MG routine slot.	ERBMFISP	
The EPSEDT address is also stored into the STSGT (so that RMF termination can free it) and into the STMEDAD of the STSMA (for use by the interval MG routine – ERBMFDSP). The length of the EPSEDT is stored into the STSGT to be used later for freeing.		

Diagram 24. Page/Swap Dataset Activity Initialization (ERBMFISP) (Part 5 of 6)

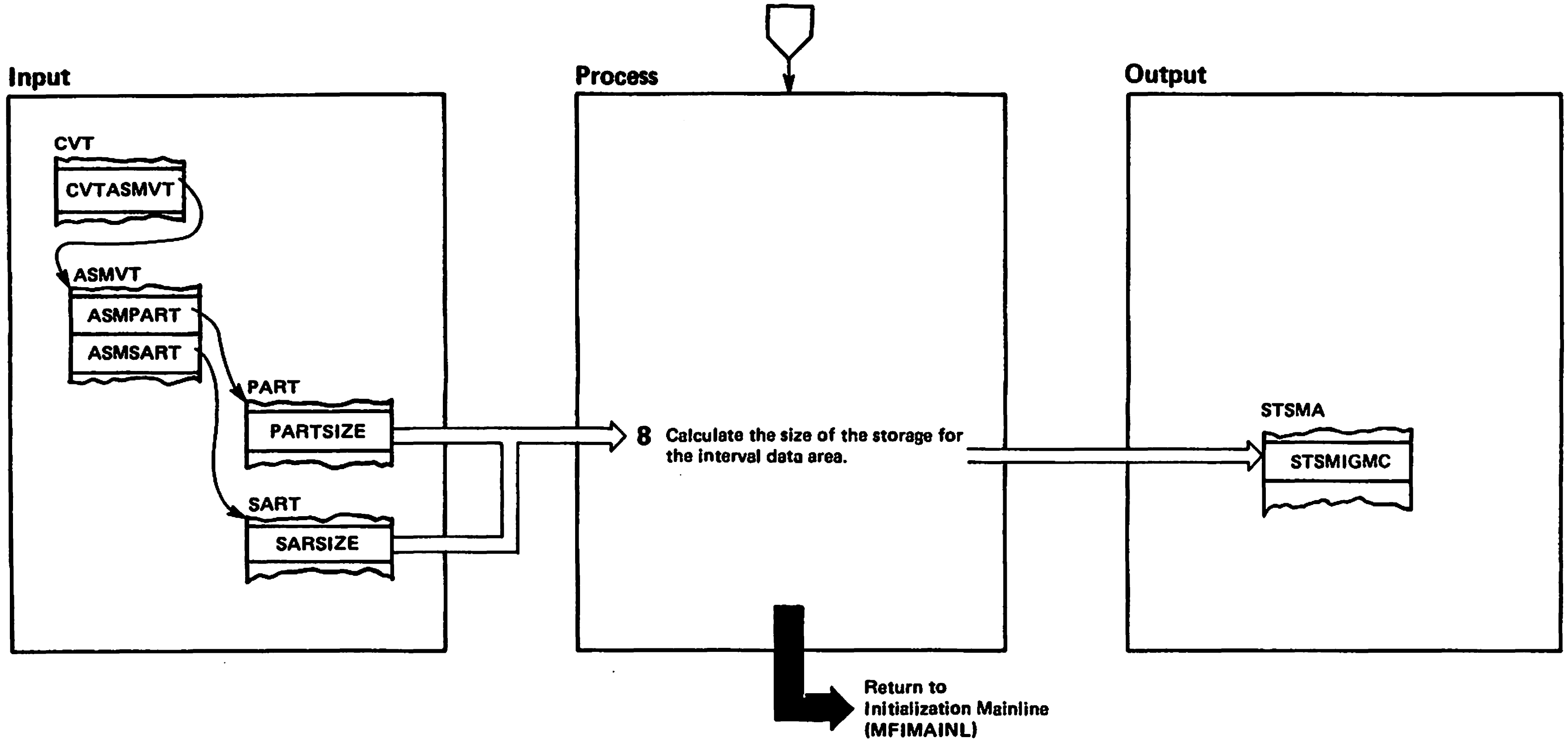


Diagram 24. Page/Swap Dataset Activity Initialization (ERBMFISP) (Part 6 of 6)

Extended Description	Module	Label
8 ERBMFISP calculates the length of the required interval data area and stores the value in the STSMA (STSMIGMC).	ERBMFISP	

The storage length for page/swap dataset data is:

4 + [4*(PARTSIZE + SARSIZE)] + 4 + [(PARTSIZE + SARSIZE) * (length of SMF75HDR + length of SMF75PRO + length of SMF75PSD)].

This formula allows for a control word, a vector table of addresses for the SMF records, a word of zeroes following the table, and storage for the maximum number of SMF records (one per page dataset and one per swap dataset).

ERBMFISP returns with a code of 12 to indicate that sampling is to be done.

ERBMFISP returns with a code of 0 if the option was not specified to sample page/swap dataset activity.

Diagram 25. Enqueue Initialization (ERBMFIEQ) (Part 1 of 10)

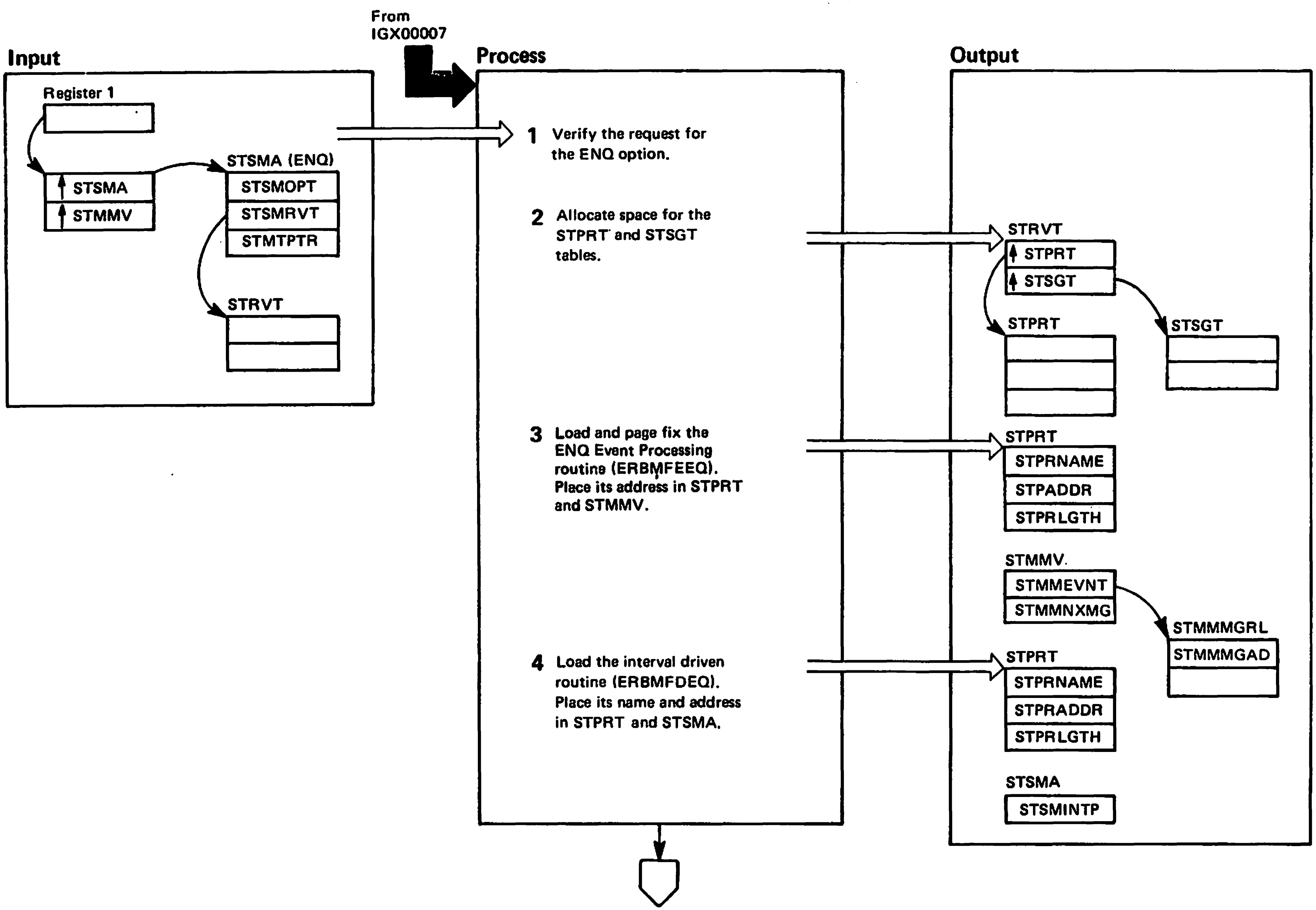
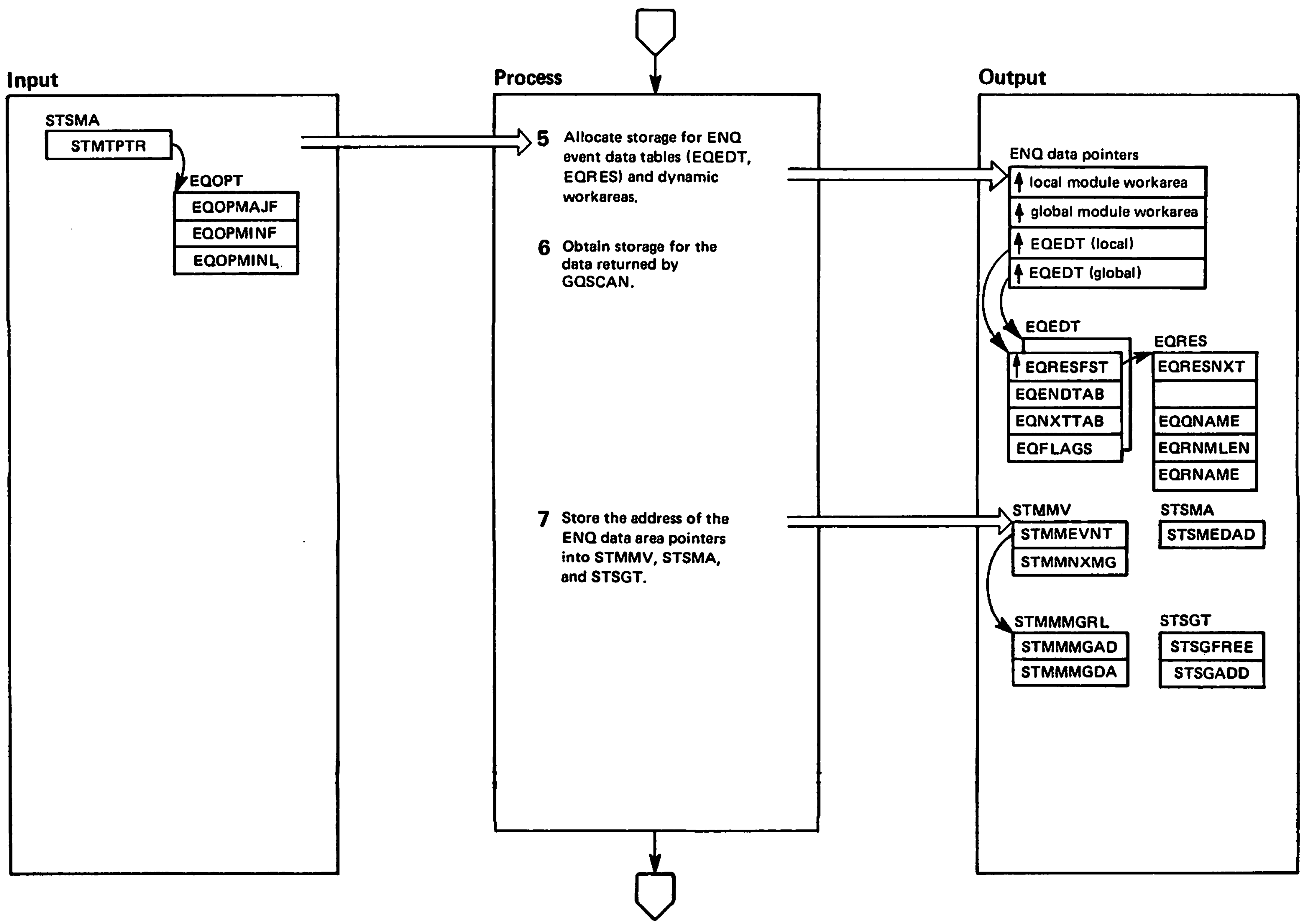


Diagram 25. Enqueue Initialization (ERBMFIEQ) (Part 2 of 10)

Extended Description	Module	Label
The Enqueue Initialization Routine initializes the data areas that RMF uses to collect ENQ data. It issues the GQSCAN macro instruction to gather the data. IGX00007 invokes ERBMFIEQ.		
1 The Enqueue Initialization routine checks the STSMOPT word of the supervisor measurement area (STSMA) for the ON condition. If ON, continue with step 2; if OFF, the user did not request the ENQ option. Return to IGX00007 with a return code of zero.	ERBMFIEQ	ERBMFIEQ
2 Issue a GETMAIN macro instruction to obtain storage in subpool 0 for the resource vector table (STRVT), the program resource table (STPRT), and the storage resource table (STSGT). STPRT contains the name, address, and the length of all programs loaded or fixed for an RMF measurement option, the first of which is ERBMFIEQ. STSGT contains the addresses and lengths of storage that ERBMFIEQ obtained. The general termination routine (ERBMFTMA) uses the tables to release these resources.		
3 Load the module ERBMFEEQ into virtual storage, and enter the Page Fix routine (PGSER) to page fix ERBMFEEQ. Save the address of ERBMFEEQ in STPRT and STMMV.		
4 Load the module ERBMFDEQ into virtual storage. Save the module's name and address in STPRT and STSMA.		

Diagram 25. Enqueue Initialization (ERBMFIEQ) (Part 3 of 10)



Extended Description	Module	Label
5 Issue a GETMAIN macro instruction to obtain storage in SQA for the enqueue data collection areas (EQEDT and EQRES) and dynamic workareas. Establish addressability to the collection areas and the workareas in SQA and initialize collection area boundaries and pointers. There is a local enqueue data collection area and a local dynamic workarea for local requests and a global enqueue data collection area and a global dynamic workarea for global requests. If a report on a specific major, minor name was requested, the storage obtained by the GETMAIN macro instruction is large enough for only one EQRES for local and global. If a specific major name was requested, the storage obtained by the GETMAIN macro instruction is approximately half the size of the storage obtained when there are no specific requests.		
6 Issue the GETMAIN macro instruction to obtain storage for a buffer to hold data returned from GQSCAN.		GETBUF
7 Store the address of the enqueue event data table (EQEDT) in STMMMGRL of the MFROUTER measurement vector table (STMMV) for the ENQ Event Handler (ERBMFEEQ) to use when an ENQHLD or ENQRLSE sysevent occurs. Also store the EQEDT address in the storage resource table (STSGT) and the supervisor measurement area (STSMA).		

Diagram 25. Enqueue Initialization (ERBMFIEQ) (Part 5 of 10)

Input

STGST
- STGLUGPT
- STGLULPT

STPRT
- STPRNAME
- STPADDR
- STPRLGTH

Process

8 Prepare for ERBMFEEQ to receive control.

a. Serialize with Monitor III.

b. Obtain storage for the ERB3GLUE workarea.

c. Load and page fix ERB3GLUE and place its address in STPRT.

d. Store the address of ERBMFEEQ in the STGST.

e. Store the address of ERB3GLUE in the CVT.

f. Release serialization with Monitor III.

Output

STGST
- STGLUGPT
- STGLULPT
- STGLUELE

STPRT
- STPRTNAME
- STPRTADDR
- STPRTLGTH

STGST
- STGSSM1EH

CVT
- CVTRMFPT

Extended Description	Module	Label
8 Prepare for ERBMFEEQ to receive control.		
a. Issues an ENQ for the resource major name RMFENQUE and the minor name 'ENQ.NOTIFY.INTER'.		
b. Issues a GETMAIN macro instruction to obtain storage in the extended SQA for a local workarea (for local requests) and a global workarea (for global requests). The ERB3GLUE module uses these workareas when an ENQHLD or an ENQRLSE sysevent occurs. Store the address and length of the workarea in the STGST. These working storage areas are obtained only if the pointers in the STGST are empty.		
c. The ERB3GLUE module coordinates ENQ event processing for RMF Monitor I and Monitor III. ERBMFIEQ loads ERB3GLUE into virtual storage, obtains a local lock, page fixes ERB3GLUE, and releases the local lock.		
d. Stores the address of ERBMFEEQ in the STGST, enabling ERB3GLUE to notify RMF Monitor I when a change in contention occurs.		
e. Stores the address of ERB3GLUE in the CVT, enabling global resource serialization to notify RMF when a change in contention occurs.		
f. Issues a DEQ to release serialization for the resource major name RMFENQUE and the minor name ENQ.NOTIFY.INTER.		

Diagram 25. Enqueue Initialization (ERBMFIEQ) (Part 7 of 10)

Input

EQOPT
- EQOPDETF
- EQOPMINF
- EQOPMAJF
- EQOPMAJN
- EQOPMINL
- EQOPMINN

EQEDT
- EQNXTTAB
- EQDATAFL
- EQENDTAB
- EQRESFST
- EQDETCNT

EQOPT
- EQOPMAJF
- EQOPMAJN
- EQOPMINF
- EQOPMINL
- EQOPMINN

ISGRIB
- RIBQNAME
- RIBRNMLN
- RIBRNAME
- RIBSTEP
- RIBSYSS
- RIBSYS
- RIBNRIBE
- RIBESTAT
- RIBETYPE
- RIBTRIBE
- RIBNTWE
- RIBNTWS
- RIBNTO
- RIBVLEN
- RIBEJBNM
- RIBEASID
- RIBESYSN

Process

9 Obtain current resource contention data. Perform the following steps, first for global resources and then for local resources.

a. Serialize with global resource serialization.

b. Obtain contention data.

c. Move user options into the EQEDT.

d. Build an EQRES with the contention data for any resource that the user requested.

e. Release the serialization with global resource serialization.

Output

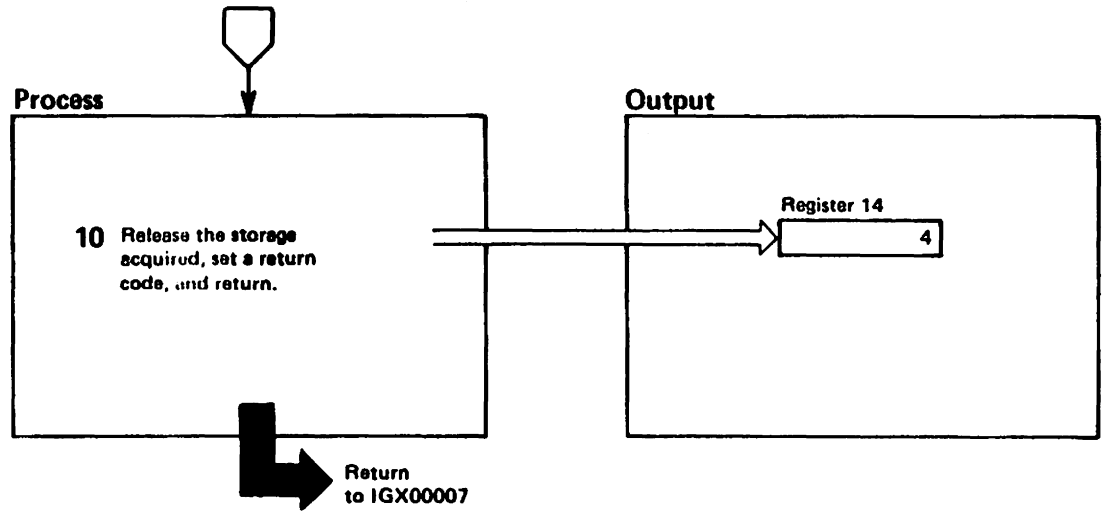
EQEDT
- EQDETAIL
- EQRNMOPT
- EQQNMOPT
- EQEDTQNM
- EQEDTRNL
- EQEDTRNM

EQEDT
- EQDATAFL
- EQRESFST
- EQNXTTAB

EQRES
- EQRESNXT
- EQWAITCT
- EQOWNCNT
- EQHLDT
- EQQNAME
- EQRNMLEN
- EQRNAME
- EQSYSS
- EQSYS
- EQMAXEXL
- EQMAXSHR
- EQMINEXL
- EQMINSHR
- EQHLDCNT
- EQTOTEVN
- EQMINHLD
- EQOWNID2
- EQWATID2
- EQWAIT1E
- EQWAIT2E
- EQOWNERE
- EQOWNER1
- EQOWNID1
- EQOWN1SY
- EQOWNER2
- EQOWN2SY
- EQWAITR1
- EQWATID1
- EQWAT1SY
- EQWAITR2
- EQWAT2SY

Extended Description	Module	Label
9 Initialize the resource contention data with data from the current global resource serialization queues. Perform the following steps once for global data and again for local data.		
a. Serialize the initialization of the resource contention data. For a global GQSCAN, obtain the local lock of the global resource serialization address space. For a local GQSCAN, obtain the CMSEQDQ and local locks.		
b. Issue the GQSCAN macro instruction. GQSCAN returns information for each resource in a resource information block (RIB), which describes the resource, and resource information block extents, (RIBEs), which describe each owner. Issue GQSCAN as many times as necessary to obtain all of the current data.		
c. Check to see if the EQDETAIL bit is on. If the bit is on, an ENQ detail report is provided; if the bit is off, a summary report is provided. If the EQQNMOPT bit is on, a report on the user-specified qname is provided. If the EQRNMOPT bit is on, a report on the user-specified rname is provided.		
d. Examine each RIB returned by GQSCAN and create an ENQ event resource contention table (EQRES) for it in the ENQ data collection area. The EQRES contains information about a requested resource. If the user requested data for a particular resource name, collect contention information only for that resource.		
e. Release any locks obtained in step 9a. ERBMFEEQ can now process contention data for the type of resource (global or local) just processed.		

Diagram 25. Enqueue Initialization (ERBMFIEQ) (Part 9 of 10)



Extended Description	Module	Label
10 Issue the FREEMAIN macro instruction to release the storage obtained for the GQSCAN buffer area. Set the return code to 4 to indicate ENQ activity is active and return to the caller.		

ERBMFIOQ - MODULE DESCRIPTION

DESCRIPTIVE NAME: I/O Queuing Initialization

FUNCTION:
Initializes the data areas that RMF requires to control the collection of I/O queuing data.

ENTRY POINT: ERBMFIOQ

PURPOSE: See function

LINKAGE: BALR - from IGX00007

CALLERS: IGX00007 - initialization mainline (MFIMAINL)

INPUT:
```
   Parameter 1 - Pointer to the supervisor state
                 measurement area (STSMA)
   Parameter 2 - Pointer to the MFROUTER measurement
                 vector tables (STMMV)
   Parameter 3 - Cycle value
```

OUTPUT:
```
   The following control blocks/tables are
   created and initialized:
   STPRT     - Program resource table
   STSGT     - Storage resource table
   STLET     - Listen event table
   EIOQED    - I/O queuing event data table
   EIOQDBs   - I/O queuing data blocks
```

EXIT NORMAL: Returns via BR 14 to caller.

EXIT ERROR:
ABEND completion code U1204 - unexpected return code from configuration retrieve module (ERBCNFGR)

EXTERNAL REFERENCES:

ROUTINES:
```
   ERBCNFGR - Retrieve configuration information
   ERBMFIQA - Initialize generation of hardware
              measurements
   IECVMAP  - IOSMAP interface
```

CONTROL BLOCKS:
```
   CVT      - Communication vector table
   ERBIOML  - RMF I/O measurement level constants
   ERBIOQDB - I/O queuing data block
   ERBIOQED - I/O queuing event data table
   ERBMFOCB - Option control block
   ERBSTGST - RMF global supervisor table
   ERBSTLET - Listen event table
   ERBSMF78 - I/O queuing SMF record:
              Subtype 1 for 308x and 4381 processors
              Subtype 3 for IBM 3090 processors
   ERBSTMMV - MFROUTER measurement vector table
   ERBSTPRT - Program resource table
   ERBSTRVT - Resource vector table
   ERBSTSGT - Storage resource table
   ERBSTSMA - Supervisor measurement area
   IECDIOCM - I/O communication area
   IECDIOSB - I/O supervisor block
   IEFUCBOB - Unit control block
   IHAICHPT - Installed channel path table
   IHAPSA   - Prefixed save area
   IHASCHIB - Subchannel information block
   IOSDMAP  - IOSMAP parameter list
```

ERBMFIOQ - MODULE DESCRIPTION (Continued)

IRACMCT - Channel measurement control table
IRARMCT - SRM control table

TABLES: IQLSTBVC - Internal LCU number selection table

ERBMFIOQ - MODULE OPERATION

1. Verifies the request for I/O queuing.

2. Obtains SPO (really 252) storage for the storage resource table (STSGT) and the program resource table (STPRT), then connects them to the resource vector table (STRVT).

3. Loads the MFROUTER service module (ERBMFEVT) into the link pack area and page-fixes it.

4. Loads the event-driven measurement gathering routine ERBMFEOQ (for 308x or 4381 processors) or ERBMFEGQ (for IBM 3090 processors) into the link pack area and page-fixes it.

5. Loads the interval measurement gathering routine ERBMFDOQ (for 308x or 4381 processors) or ERBMFDGQ (for IBM 3090 processors).

6. Loads ERBCNFGR into the link pack area and page-fixes it, then calls it to obtain the number of logical control units (LCUs) in the system.

7. Obtains storage for an internal LCU number table (IQLSTBVC) to temporarily hold information for all LCUs selected, either by LCU number or by device class, and counts the number of selected LCUs. For these LCUs, sets up information provided by module ERBCNFGR and by the IOSLOOK macro instruction.

8. Allocates global fixed storage (SP 245) for the I/O queuing control blocks and initializes them with information from the previous step, information provided by ERBCNFGR and the IOSINFO macro instruction, and information from the installed channel path table. The control blocks are:

   - EIOQED - I/O queuing event data table (when RMF is running on a processor, the EIOQED includes EIOQ3090).
   - EIOQDB - I/O queuing data blocks (when RMF is running on a processor, the EIOQDB includes the data block extension, EIOQDBX).

9. Frees the storage obtained for the temporary internal LCU number table (IQLSTBVC).

10. Obtains global fixed storage (SP 245) for an IOSB and a SCHIB to be used by this module for the initial path connectivity check and by modules ERBMFEOQ and ERBMFDOQ for the IOS/STSCH interface, places the address of the obtained area into the STSGT and the addresses of the control blocks into the event data table (EIOQED), and initializes common IOSB fields.

11. Loops through all LCUs. For each device within an LCU, issues an IOSMAP request, checks

for online channel paths that have no connectivity to any device in the LCU, sets up the path connectivity information in the I/O queuing data blocks, and initializes the channel path online mask (EIOQCPOM).

12. Calculates 'N-cycle' value and places it into the EIOQED. The 'n-cycle' value indicates the number of cycles that elapse between each retrieval of model-dependent data. RMF retrieves this data by means of the IOS/STSCH interface (for 308x and 4381 processors) or the DIAGNOSE interface (for processors).

13. Calls module ERBMFIQA to initialize generation of hardware measurements and additional fields in the I/O queuing control blocks.

14. Obtains SP0 (really 252) storage for the listen event table (STLET), places its address in the resource vector table (STRVT), and creates entries for listen exits: ERBLXIOQ, ERBLXVCP, and, if RMF is running on an IBM 3090 processor, ERBLXCMF. IGX00007 activates the listen exits.

15. Calculates the size of the interval data area for the SMF record image (SMF record 78, subtype 1 for 308x and 4381 processors or subtype 3 for IBM 3090 processors).

16. Returns to the caller.

RECOVERY OPERATION:

Module ERBMFSDE gets control if any abnormal termination occurs.

ERBMFIOQ - DIAGNOSTIC AIDS

ENTRY POINT NAME: ERBMFIOQ

MESSAGES: None

ABEND CODES:

U1204 - Unexpected return code from configuration
        retrieve module (ERBCNFGR)

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

```
Register 15 contains one of the following:
'00'X - No I/O queuing activity report
        requested, or
'40'X - Hardware measurement generation
        could not be activated
'2C'X - A composite code that consists of the
        following:
        '04'X - Activate MFROUTER
        '08'X - Enqueue TQE
        '20'X - Activate listen exits
```

REGISTER CONTENTS ON ENTRY:

```
Register  1 - Parameter list address
Register 13 - Save area address
Register 14 - Return address
Register 15 - Entry point address
```

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

```
Register  0 - 14  Restored to original values
Register  15      Return code
```

ERBMFIOQ - I/O Queuing Initialization

STEP 01

IGX00007 - initialization mainline (MFIMAINL)

PARAMETERS

ERBMFIOQ

IQSMAVP IQMMVVP IQCYCVVF

Initializes the data areas that RMF requires to control the collection of I/O queuing data.

01 Verifies that the I/O queuing option has been specified by checking the option status flag in the STSMOPT word of the supervisor measurement area (STSMA).

If the option status bit indicates that I/O queuing has not been requested, ERBMFIOQ clears the option word and returns to the caller with a return code of zero. Otherwise, I/O queuing was requested, and processing continues.

02 Builds and initializes storage and program resource tables.

A. Allocates storage for the storage resource table, connects it to the resource vector table, stores the subpool and length in the first word of the obtained area and sets up the current index.

B. Allocates storage for the program resource table, connects it to the resource vector table, stores the subpool and length in the first word of the obtained area and sets up the current index.

03 Loads and page-fixes all modules needed to generate an I/O queuing activity report.

A. Loads the MFROUTER service module, ERBMFEVT, and places its address into the CVT. Stores its name, address, and length in the program resource table, and updates the index to the next entry. The length is stored as a negative number to indicate that a page-free is needed if any abnormal termination occurs.

B. Page-fixes module ERBMFEVT with TCB=0 to keep it fixed even when RMF is swapped out.

CVT

CVTMFRTR

ERBIOML

IOML308X IOML4381

C. Depending on the processor type, loads the cycle-driven gathering module ERBMFEOQ (for 308x or 4381 processors) or ERBMFEGQ (for processors).

D. Stores the name, address, and length of the cycle-driven gathering module into the program resource table and updates the index to the next entry. The length is stored as a negative number to indicate that a page-free is needed if an abnormal termination occurs.

E. Page-fixes the cycle gathering module ERBMFEOQ or ERBMFEGQ with TCB=0 to keep it fixed even when RMF is swapped out.

F. Stores the module address and name into the MFROUTER vector table for use while processing a timer event.

ERBIOML

IOML308X IOML4381

G. Depending on the processor type, loads the interval-driven MG routine ERBMFDOQ (for 308x and 4381 processors) or ERBMFDGQ (for processors).

H. Stores the name, address, and length of the interval-driven MG routine into the program resource table, updates the index to the next entry, and saves the address and name in the STSMA.

I. Loads the configuration retrieve module, ERBCNFGR

J. Stores the name, address, and length into the program resource table and updates the index to the next entry. The length is stored as a negative number to indicate that a page-free is needed if an abnormal termination occurs.

K. Page-fixes ERBCNFGR with TCB=0 to keep it fixed even when RMF is swapped out.

04 Builds the internal LCU number table.

A. Calls ERBCNFGR to obtain the number of LCUs in the system. If ERBCNFGR issues an unexpected return code, requests an ABEND with user code 1204.

ERBCNFGR
IQFCT4CC, IQNLCUVF

B. Calculates the amount of storage needed for the internal LCU number selection table, obtains the storage, clears the storage and sets the count of selected LCUs to zero.

05 For each LCU in the system, builds an LCU number table entry. Each entry consists of a device number, the associated UCB pointer, and the device class.

A. Calls ERBCNFGR to obtain the first device number and the number of devices in this LCU. If ERBCNFGR issues an unexpected return code, marks the LCU as if no UCB has been found.

ERBCNFGR
IQFCT5CC, IQLCUNVC, IQDVNNVF, IQNDEVVF

B. Issues an IOSLOOK macro instruction to get the associated UCB address.

C. If IOS could not return a UCB address, ERBMFIOQ calls ERBCNFGR and issues IOSLOOK again to retry with other device numbers in this LCU.

ERBCNFGR
IQFCT3CC, IQLCUNVC, IQLSDEVN(IQLCUXVF), IQDVNNVF

UCB

UCBDVCLS

D. If a UCB exists for any device in this LCU and if its device class has been selected on the IOQ option, marks this LCU as selected and increments the count of selected LCUs.

OCB

OCBDBEL

OCB

OCBNEXT OCBADDR
OCBNUM1 OCBNUM2

E. If the user specified the IOQ NMBR suboption, ERBMFIOQ loops through the OCB chain, marks the specified LCUs as selected, and increments the count of selected LCUs for each LCU that has not previously been counted.

06 For the selected LCUs, builds the I/O queuing event data structure, which consists of the event data table (EIOQED) and the data blocks (EIOQDBs).

ERBIOQDB

EIOQDB EIOQDBX

ERBIOML

IOML3090

A. Calculates the amount of storage needed for the I/O queuing data block structure and obtains global fixed (SP245), key zero storage.

B. Saves the address of the area in the STSMA and clears the area.

ERBMFIOQ - I/O Queuing Initialization STEP 06C

C. Stores I/O queuing event data area information into the storage resource table for reference and freeing by the termination routine and also saves the area address in the MFROUTER table and in the STGST.

ERBIOML: IOML3090
IRACMCT: CMCTPAA

07 Initializes the EIOQED (I/O queuing event data table).

08 Initializes one EIOQDB (I/O queuing data block) for each selected LCU.

A. Sets the logical control unit number.

ERBIOQDB: EIOQLCUN

B. Moves the device number and, if a UCB exists, the UCB pointer and the device class. Issues the IOSINFO macro instruction to obtain the associated subchannel number. If no UCB exists or if no subchannel number could be obtained, turns on the no UCB available flag (EIOQNUCB).

ERBIOQDB: EIOQDVCL, EIOQNUCB, EIOQUCBA, EIOQDNUM

C. Calls ERBCNFGR to retrieve configuration information (channel path IDs and physical control units). If ERBCNFGR issues an unexpected return code, requests an ABEND with user code 1204.

ERBCNFGR
IQFCT6CC, IQLCUNVC, IQCNFDV

ERBIOQDB: EIOQDCNT

ERBIOQDB: EIOQCPID
IHAICHPT: ICHCONFG ICHONLIN
UCB: UCBCHPID
ERBIOQDB: EIOQCHPN

D. Initializes each configuration data section. Moves channel path IDs and associated control unit numbers, initializes the installed/online status flags, and associates a channel path ID with its position in the logical path mask (LPM).

ERBIOQDB: EIOQCPID, EIOQCPI, EIOQCPO, EIOQCUN, EIOQCU, EIOQCPMX

ERBIOML: IOML308X IOML4381

E. Sets the no base values available flag (EIOQMDDB) for 308x and 4381 processors and the no hardware measurements activated flag (EIOQNHMA) for all processor types.

ERBIOQDB: EIOQMDDB, EIOQNHMA

ERBMFIOQ - I/O Queuing Initialization STEP 08F

ERBIOML
IOML3090
ERBIOQDB
EIOQDB EIOQDBX

F. If RMF is running on a processor, initializes the pointer to the data block extension (EIOQDBX).

ERBIOQDB
EIOQDBX

09 Frees the storage used for the temporary internal LCU number selection table.

10 Builds and initializes the IOSB and the SCHIB to be used by the IOS/STSCH interface.

ERBIOML
IOML308X IOML4381

A. Obtains storage for the IOSB and the SCHIB and connects it to the storage resource table (STSGT).

IOSB

B. Saves the pointers to the IOSB and the SCHIB in the EIOQED, zeroes the obtained storage area, and initializes common fields in the IOSB.

IOSB
IOSASID
IOSSYN
IOSSCHIB

ERBIOQDB
EIOQDCNT

11 Processes each I/O queuing data block to perform an initial pass connectivity check. The check consists of issuing an STSCH request for all devices in an LCU for which a UCB exists. Uses the results to set up path connectivity status flags and to initialize the channel path online mask.

ERBIOQDB
EIOQNUCB
ERBIOQDB
EIOQUCBA
IOSDMAP

A. For each device in an LCU, uses the IOSMAP interface to retrieve the logical path mask (LPM) and check the path connectivity for this device.

IOCMAPE

IOSDMAP
MAPUCB

ERBIOQDB
EIOQCPI EIOQCPMX
IOSDMAP
MAPLPM
ERBIOQDB
EIOQCHPN

B. Calls ERBCNFGR to obtain the device number of the next device in the same LCU. If ERBCNFGR issues an unexpected return code, requests an ABEND with user code 1204.

ERBCNFGR
IQFCT3CC, EIOQLCUN, EIOQDNUM, IQDVNNVF

ERBIOQDB
EIOQDB EIOQCPO
ERBIOQDB
EIOQCPOM EIOQCPID
EIOQCHPN

C. Issues IOSLOOK to obtain the UCB address associated with the new device number.

ERBIOQDB
EIOQDNUM

D. If the request fails, continues with the next device in the LCU. If the request succeds, saves the device number and its UCB pointer.

ERBIOQDB
EIOQUCBA

E. Sets the path connectivity flags in the EIOQDB and initializes the channel path online mask (EIOQCPOM).

ERBIOQDB
EIOQCPOM
EIOQVPOF

PARAMETERS
IQCYCVVF

12 Calculates the 'N-cycle' value and places it into EIOQNCYC. (The 'N-cycle' value indicates the number of cycles that elapse between each retrieval of model-dependent data.)

13 Calls module ERBMFIQA to activate the generation of hardware measurements. If measurements could not be activated (non-zero return code received from module, ERBMFIQA) ERBMFIOQ sets its return code to x'40' and returns to the caller.

ERBMFIQA
IQSTRTCC, STGSIOQ

14 If hardware measurements have been activated, builds the I/O queuing listen event table.

ERBIOML
IOML3090
ERBSTLET
STLENTRY

A. Obtains storage for the listen event table, connects it to the resource vector table (STRVT), zeroes the obtained storage, and initializes the table header.

ERBSTLET
STLENAME
STLESP
STLELEN

ERBSTLET
STLEQUAL STLEQMSK

B. Sets up a listen event table entry for ERBLXIOQ.

ERBSTLET
STLECODE
STLEQUAL
STLEQMSK
STLEENAM

ERBSTLET

STLEQUAL STLEQMSK

ERBIOML

IOML3090

ERBSTLET

STLEQUAL STLEQMSK

ERBIOML

IOML308X IOML4381

ERBSMF78

SMF78HDR SMF78PRO
R781CS R781DS
R783GD R783IQD
R783CS R783DS

C. Sets up a listen event table entry for ERBLXVCP.

D. If RMF is running on an IBM 3090 processor, sets up a listen event table entry for ERBLXCMF.

15 Depending on the processor type, places the length and subpool of the interval data area into the STSMA.

16 Sets the composite return code of X'2C' to indicate all of the following:

- enable MFROUTER (X'04')
- enqueue the system TQE (X'08')
- activate listen exits (X'20')

17 Returns to the caller.

ERBSTLET

STLECODE
STLEQUAL
STLEQMSK
STLEENAM

ERBSTLET

STLECODE
STLEQUAL
STLEQMSK
STLEENAM

Diagram 27. Virtual Storage Initialization (ERBMFIVS) (Part 1 of 8)

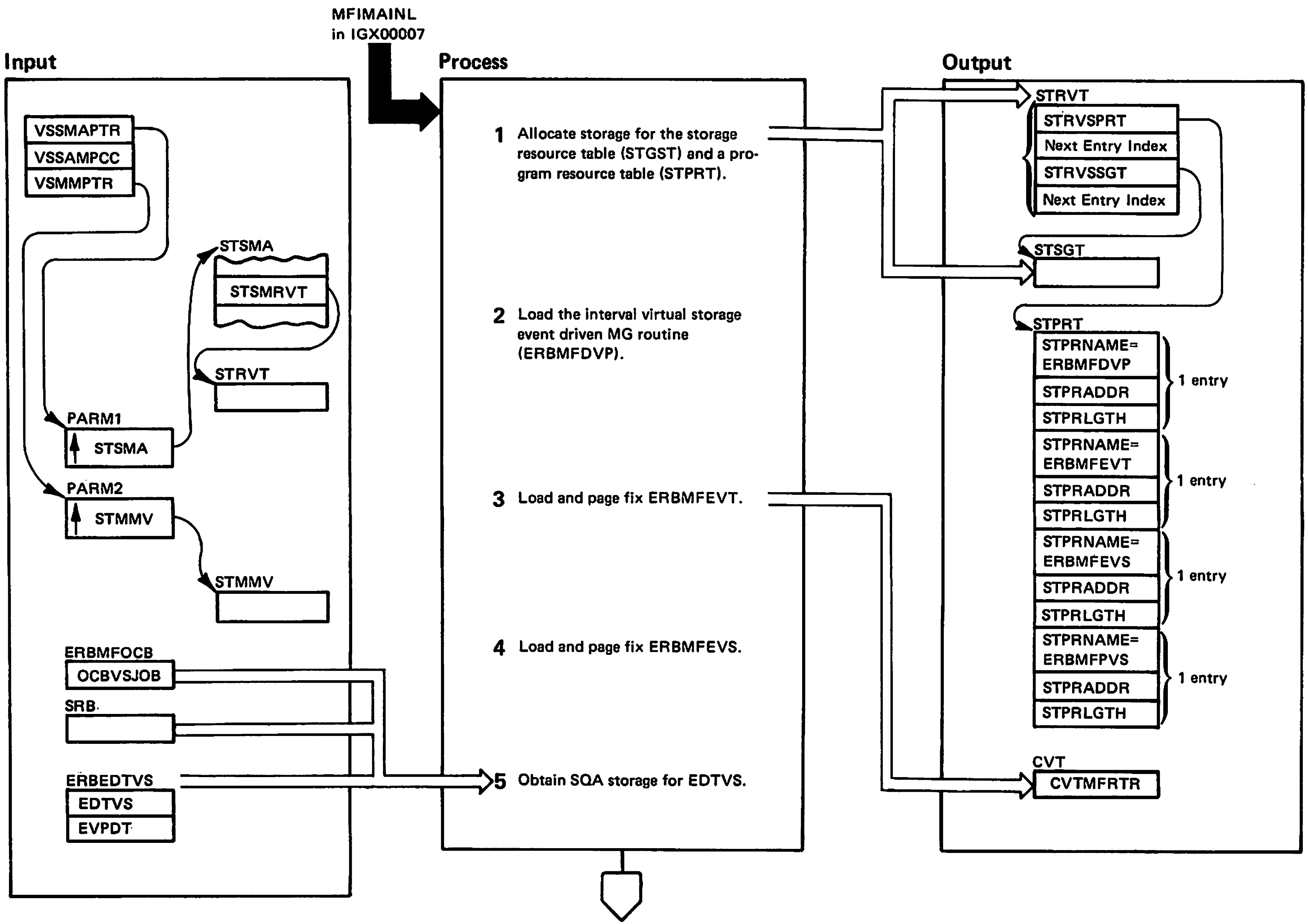


Diagram 27. Virtual Storage Initialization (ERBMFIVS) (Part 2 of 8)

Extended Description	Module	Label

ERBMFIVS, which is called by MFIMAINL in IGX00007, obtains and initializes data areas that are used to control the collection of virtual storage data. It loads the virtual storage interval measurement gathering module, ERBMFEVS, and the MFROUTER processor, ERBMFEVT. If private area monitoring is also required, it also loads the virtual storage private area sampler, ERBMFPVS. It obtains storage for the SMF record and processes the fields that are static for the record.

1 ERBMFIVS issues a GETMAIN to obtain subpool 0 storage for the program resource table (STPRT) and the storage resource table (STSGT). ERBMFIVS then saves their addresses in the STVSPRT and STRVSSGT fields of the resource vector table (STRVT) and initializes the STRVNPRT and STRVNSGT index fields in the STRVT. The index is used in subsequent processing to step through contiguous entries in the STPRT and STSGT.

2 Loads the interval driven MG routine for virtual storage, ERBMFDVP, saves the address of ERBMFDVP in the supervisor measurement area (STSMA), and stores the name, address, and length of ERBMFDVP in the program resource table (STPRT).

3 Loads the MFROUTER processor, ERBMFEVT, saves the address in the CVT, and stores the name, address, and length of ERBMFEVS in the resource list (STPRT). ERBMFIVS issues the PGSER macro instruction to page fix ERBMFEVS, and sets the TCB equal to zero to keep the module fixed in LPA while the rest of RMF is swapped out.

Extended Description	Module	Label

4 Loads ERBMFEVS, the virtual storage sampling routine, stores the name, address, and length of ERBMFEVS in the resource list (STPRT), and stores the address of ERBMFEVS in the MFROUTER vector table (STMMV) for the timer event. ERBMFIVS issues the PGSER macro instruction to page fix ERBMFEVS. Set the TCB equal to zero to keep the module fixed in LPA while the rest of RMF is swapped out.

5 Issues the GETMAIN macro instruction to obtain SQA (subpool 245) for the virtual storage event data table (EDTVS), then zeroes the entire area and saves the address in the STSMA, STSGT, and STMMV.

Diagram 27. Virtual Storage Initialization (ERBMFIVS) (Part 3 of 8)

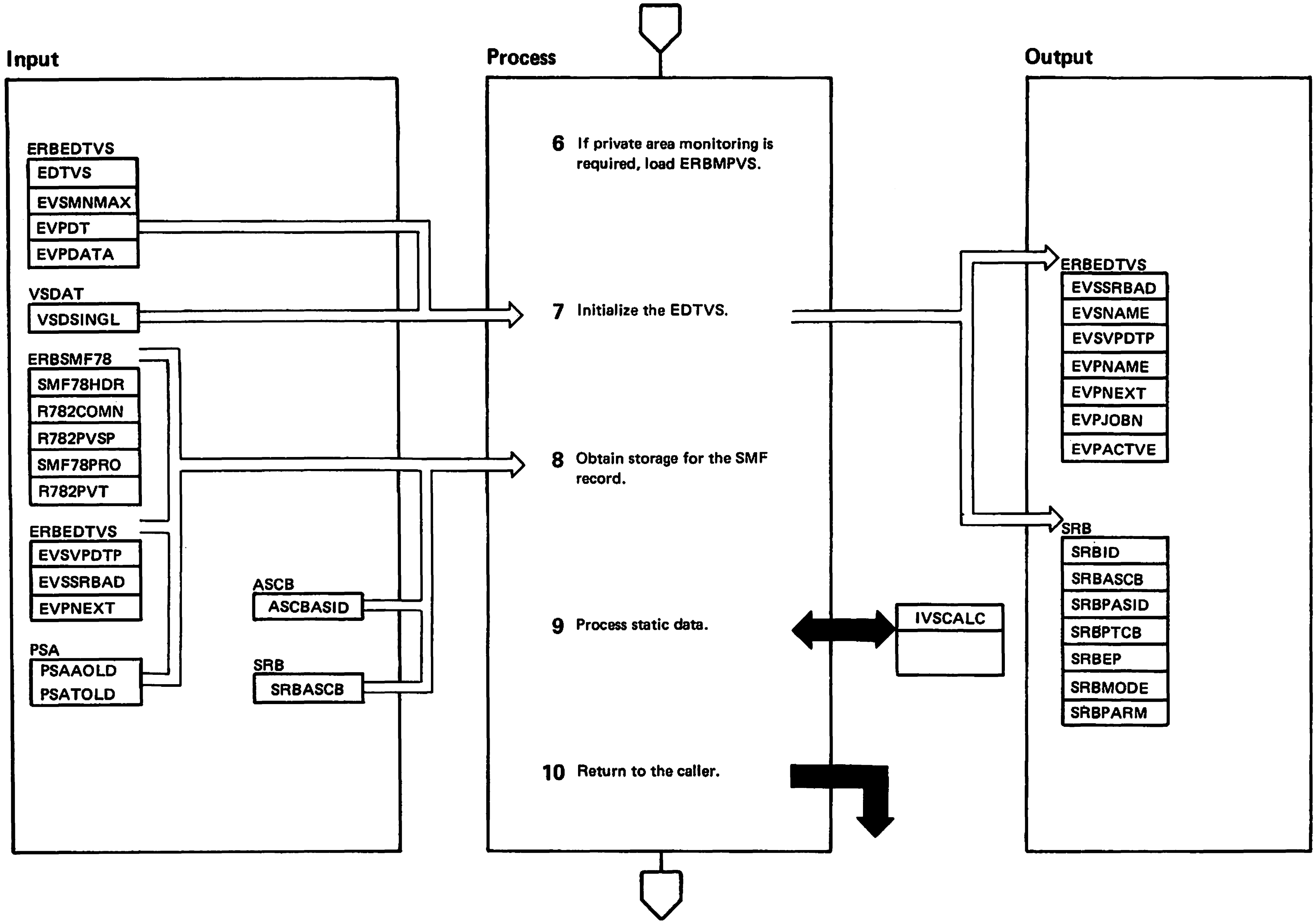


Diagram 27. Virtual Storage Initialization (ERBMFIVS) (Part 4 of 8)

Extended Description | **Module** | **Label**

6 If private area monitoring is required (the number of jobnames is not zero):

- Loads the SRB routine ERBMFPVS.
- Stores the name, address, and length of ERBMFPVS in the resource list (STPRT).
- Stores the address of ERBMFPVS in the virtual storage event data table (EDTVS).
- Issues the AXSET macro instruction for secondary addressing.
- Completes the SRB information.

7 Initializes the EDTVS. Completes the static areas of the EDTVS and EVPDT and initializes all of the minimum values for both.

8 After calculating the size of the interval data areas, issues the GETMAIN macro instruction to obtain storage for the SMF record (type 78, subtype 2). The storage length for virtual storage data is calculated using the following formula:

```
length=4+ (length of SMF78HDR) + (length
        SMF78PRO) + (length R782COMN) +
        (( (number of jobnames)* ((length
        R782PVT) + (length R782PVSP)) *145)
```

This formula allows for a prefix control word and a virtual storage SMF record. The constant 145 includes the maximum 135 user and system subpools, plus ten extra subpools for expansion.

9 Calls subroutine IVSCALC to process the static virtual storage data.

10 Returns to the caller. A return code of 12 indicates successful initialization.

Diagram 27. Virtual Storage Initialization (ERBMFIVS) (Part 5 of 8)

Input

ERBSMF78
- SMF78PRS
- SMF78DCS
- SMF78ASS
- SMF78ASN
- SMF78PRL
- SMF78DCL
- SMF78ASL

CVT
- CVTRONS CVTERWNS

ERBSMF78
- SMF78HDR R782PVT

GDA
- GDAPVTSZ GDAEPVT
- GDAEPVTS GDACSA
- GDACSASZ GDAECSA
- GDAECSAS GDASQA
- GDASQASZ GDAESQA
- GDAESQAS

CVT
- CVTPLPAS CVTPLPAE
- CVTEPLPS CVTEPLPE
- CVTRWNS CVTRWNE
- CVTERWNE CVTRONE
- CVTBLDLS CVTBLDLE
- CVTMLPAS CVTMLPAE
- CVTEMLPS CVTEMLPE
- CVTFLPAS CVTFLPAE
- CVTEFLPS CVTEFLPE

ERBSMF78
- R782PS R782EPS
- R782ENA SMF78ASS

Process

IVSCALC Subroutine

11 Complete the triplets.

12 Calculate the address and size of each unique MVS storage area.

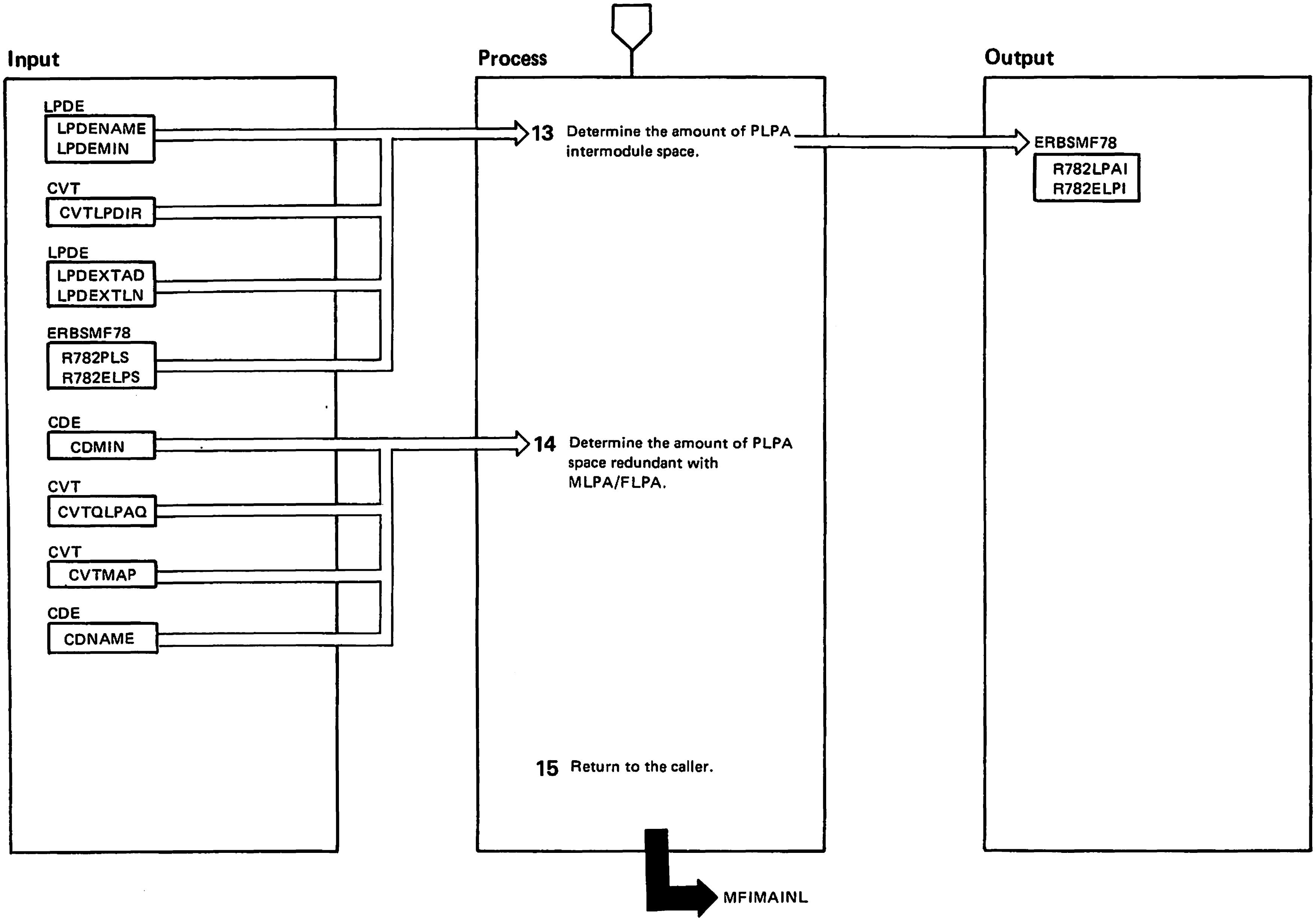
Output

ERBSMF78
- SMF78RTY
- SMF78STY
- SMF78TRN
- SMF78PRS
- SMF78PRL
- SMF78PRN
- SMF78DCS
- SMF78DCL
- SMF78DCN
- SMF78ASS
- SMF78ASL
- SMF78ASN
- SMF78SPS
- SMF78SPL

ERBSMF78
- R782PA
- R782PS
- R782EPA
- R782EPS
- R782CA
- R782CS
- R782ECA
- R782ECS
- R782PLA
- R782PLS
- R782ELPA
- R782ELPS
- R782SA
- R782SS
- R782ESA
- R782ESS
- R782MR
- R782EMR
- R782NA
- R782ENA
- R782NS
- R782ENS
- R782BA
- R782BS
- R782MLA
- R782MLS
- R782EMLA
- R782EMLS
- R782FLA
- R782FLS
- R782EFLA
- R782EFLS
- R782JOBN

Extended Description	Module	Label
IVSCALC Subroutine		
11 Completes the triplets (offset, length, and number) for the SMF record (type 78, subtype 2).		
12 Calculates the address and size of each of the unique MVS storage areas (such as the nucleus or PLPA), both above and below the 16 megabyte line.		

Diagram 27. Virtual Storage Initialization (ERBMFIVS) (Part 7 of 8)



Extended Description	Module	Label
13 Scans the LPDEs [to find the amount of PLPA intermodule space]. ERBMFIVS finds the size of the LPDE directory, the size of the storage actually used by the PLPA, and the size of the storage used by the extended PLPA (EPLPA), then subtracts the sum of these values from the PLPA size.		
14 Determine the amount of PLPA/EPLPA storage that is inaccessible because modules with duplicate names exist in the FLPA/MLPA.		
a. Scans the system CDE chain to find the names of the modules in the FLPA/MLPA.		
b. Calls IEAVVMSR, the name search routine, to scan the PLPA and EPLPA for each name in the FLPA/MLPA.		
c. Stores the total storage size of all duplicate modules in the SMF record.		
15 Returns to the subroutine caller.		

Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 1 of 12)

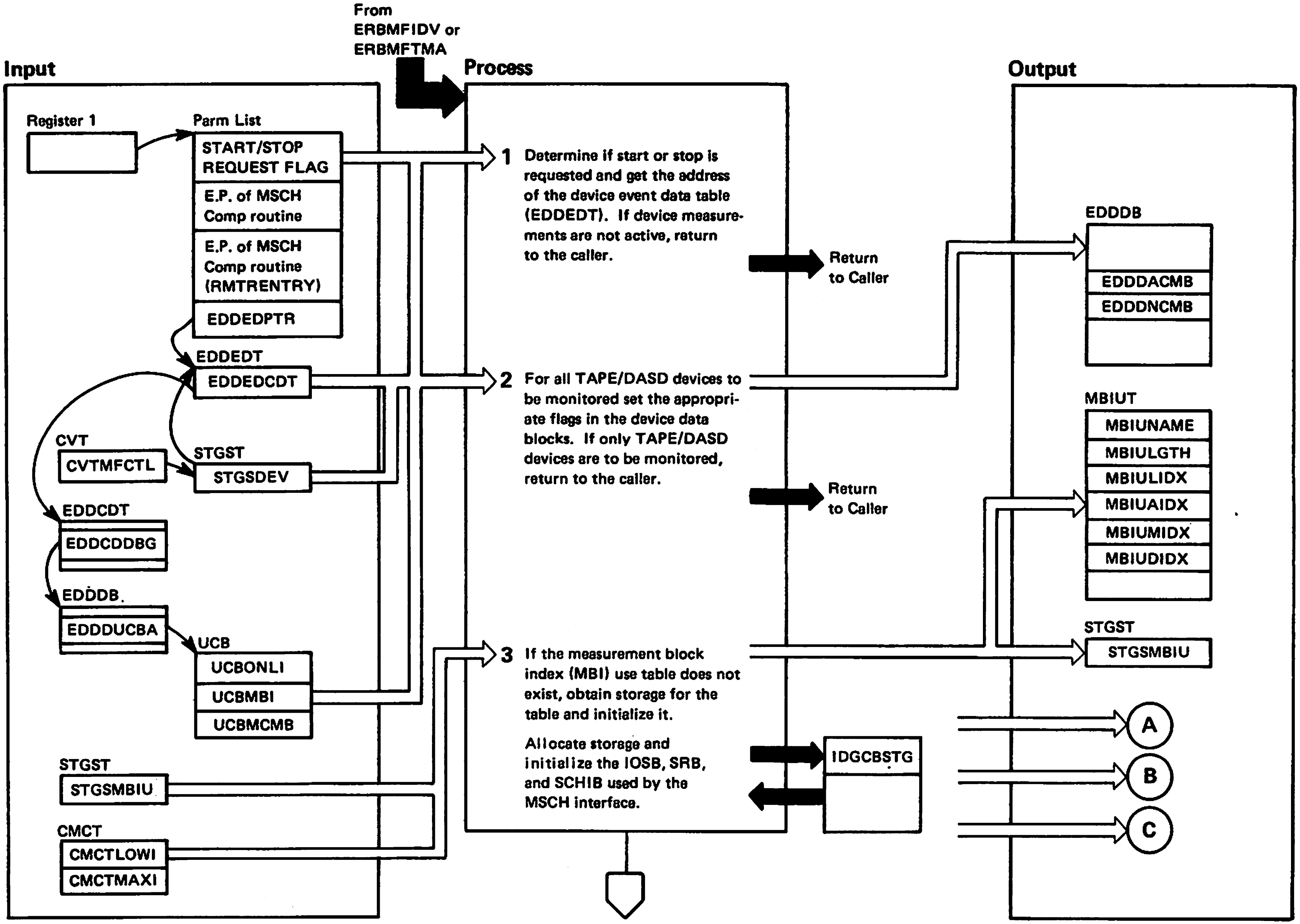


Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 2 of 12)

Extended Description	Module	Label
ERBMFIDA starts or stops hardware measurements for devices other than tape or direct access devices. For these devices the system resource manager (SRM) controls the collection of hardware measurement data. If start is specified, one CMB slot is assigned from the upper portion of the CMB to each device to be monitored and a MSCH request is issued to set the MBI and MM bits in the UCB. If stop is specified, a MSCH request is issued to turn the MBI and MM bits off.		
1 ERBMFIDA checks the first parameter to determine if start or stop processing is requested. If stop is requested, ERBMFIDA retrieves the address of the device event data control block from the RMF global storage block. The address must be present or processing is terminated. If start is requested, ERBMFIDA retrieves the address of the device event data control block from the parameter list.		
2 For all TAPE/DASD devices to be monitored, sets the appropriate flags in the channel measurement blocks (CMBs). For start, if the device is on-line and the MBI and MM values are set, sets the CMB data available flag (EDDDACMB) and resets the no CMB data available flag (EDDDNCMB). In all other cases, resets EDDDACMB and sets EDDDNCMB. Because SRM controls the gathering of hardware measurement data for these devices, no other processing is required. If only TAPE/DASD devices are monitored, returns to the caller.		
3 If the measurement block index (MBI) use table does not exist, obtains fixed global storage (ESQA) for the table and initializes it.		
Calls subroutine IDGCBSTG to obtain storage for and initialize an IOSB, an SRB, and a SCHIB for a MSCH request.		IDGCBSTG

Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 3 of 12)

Input

CVT
- CVTDCQA

DCQ
- DCQ FIRST
- DCQ DEVCL
- DCQ UCBCT
- DCQCHAIN
- DCQUCBAD

UCB
- UCBMBI
- UCBMCHB
- UCBNXUCB

Process

4 Determine if start or stop is requested.

If start go to Step 9.

5 STOP:

If an asynchronous MSCH request is pending, issue an MSCH cancel request to dequeue the request, and if the request is successfully dequeued, reset the MBI use flags.

IOCCNXLE

IDCLMBIU

6 Issue the MSCH command to turn off the measurement bits, and, if successful, reset the MBI use flags.

IOCMSCQE

7 If the previous MSCH (Step 6) was unsuccessful, issue an asynchronous MSCH command and obtain a new set of control blocks (IOSB, SRB, and SCHIB).

IOCMSCBQE

8 Free storage allocated to SCHIB, IOSB, and SRB and return to the caller.

Return to Caller

Output

MBIUT
- MBIULIDX
- MBIUAIDX
- MBIUENT

EDDDB
- EDDDIOSB

Extended Description	Module	Label
4 Checks the first parameter to determine if start or a stop was requested. For a start, continues processing at Step 9.		
5 If stop processing has been requested, the presence of an IOSB address in the device data block indicates that the completion routine for the asynchronous MSCH request was not scheduled yet. In this case, issues a MSCH cancel request to dequeue the request. If the return code (4) indicates that the request could not be found, sets a cancel flag to ensure that a MSCH request is issued to do the clean up; otherwise, calls subroutine IDCLMBIU to reset the MBI use flags.		IDCLMBIU
6 If the MBI and MM bits are set or if the IDCNLFVB cancel flag is on, initializes the SCHIB and the IOSB and issues a MSCH command to turn off the hardware measurement mode. If successful, calls subroutine IDCLMBIU to reset the MBI use flags.		IDCLMBIU
7 If the return code from the previous MSCH command is 4 (subchannel temporarily not available), initializes the appropriate control blocks and issues an asynchronous MSCH command. Places the address of the IOSB in the EDDDIOSB field of the device data block to indicate a pending asynchronous MSCH command. ERBMFIDX handles the completion of the MSCH. ERBMFIDA then calls subroutine IDGCBSTG to obtain a new set of control blocks (IOSB, SRB, and SCHIB).		IDGCBSTG
8 After all the non-TAPE/DASD devices have been processed, ERBMFIDA frees the storage allocated to the SCHIB, IOSB, and SRB and then returns to the calling routine.		

Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 5 of 12)

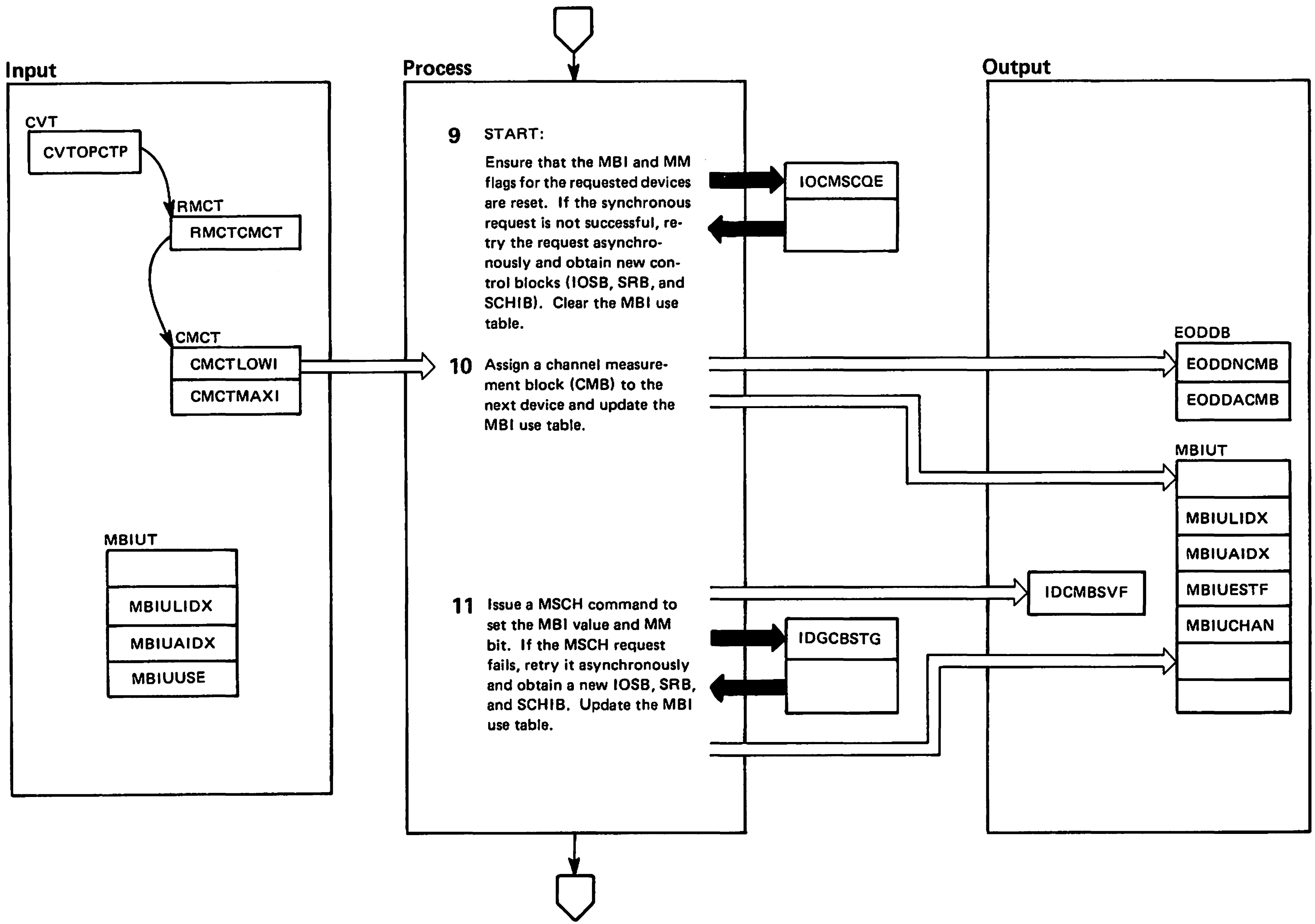


Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 6 of 12)

Extended Description	Module	Label
9 Checks all non-TAPE/DASD UCBs in the system to determine if the MBI and MM values are set in the UCB. If hardware measurements are currently being taken for a device, issues a MSCH request to turn off the MBI and MM bit settings. ERBMFIDA takes this action to ensure that no MBI value is used more than once. If the synchronous MSCH request returns code 04, retries the request asynchronously. ERBMFIDX handles the completion of the MSCH.		
Calls subroutine IDGCBSTG to obtain a new set of control blocks, (IOSB, SRB, and SCHIB). Calls subroutine IDCLMBIU to clean up the MBI use table.		IDGCBSTG IDCLMBIU
10 Assigns a 32-byte channel measurement block (CMB) slot to the non-TAPE/DASD device to be monitored by RMF. If no CMB slot is available, marks the associated EDDDB to indicate that no hardware measurements can be obtained and increments an interval 'CMB slot missing' counter; otherwise, updates the MBI use table.		
11 After ERBMFIDA has allocated the CMB slot, issues a synchronous MSCH request for the device to set up the MBI value and MM bit in the UCB and subchannel. If the request fails (return code = 4), issues an asynchronous MSCH command to retry the request and calls subroutine IDGCBSTG to obtain storage for a new IOSB, SRB, and SCHIB. In any case, updates the MBI use table.		IDGCBSTG

Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 7 of 12)

Input

IDCMBSVF

Process

12 If, after all devices have been processed, the 'missing CMB slot' counter ≠0, issue a message.

ERBMFMPR

13 Free storage allocated to SCHIB, IOSB, and SRB and return to the caller.

Return to Caller

Output

Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 8 of 12)

Extended Description	Module	Label
12 If after all devices have been processed, the internal 'CMB slot missing' counter is not equal to zero, loads the message processing module ERBMFMPR, issues error message ERB261I, and deletes ERBMFMPR.		
13 Frees storage allocated to SCHIB, IOSB, and SRB and returns to the calling routine.		

Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 9 of 12)

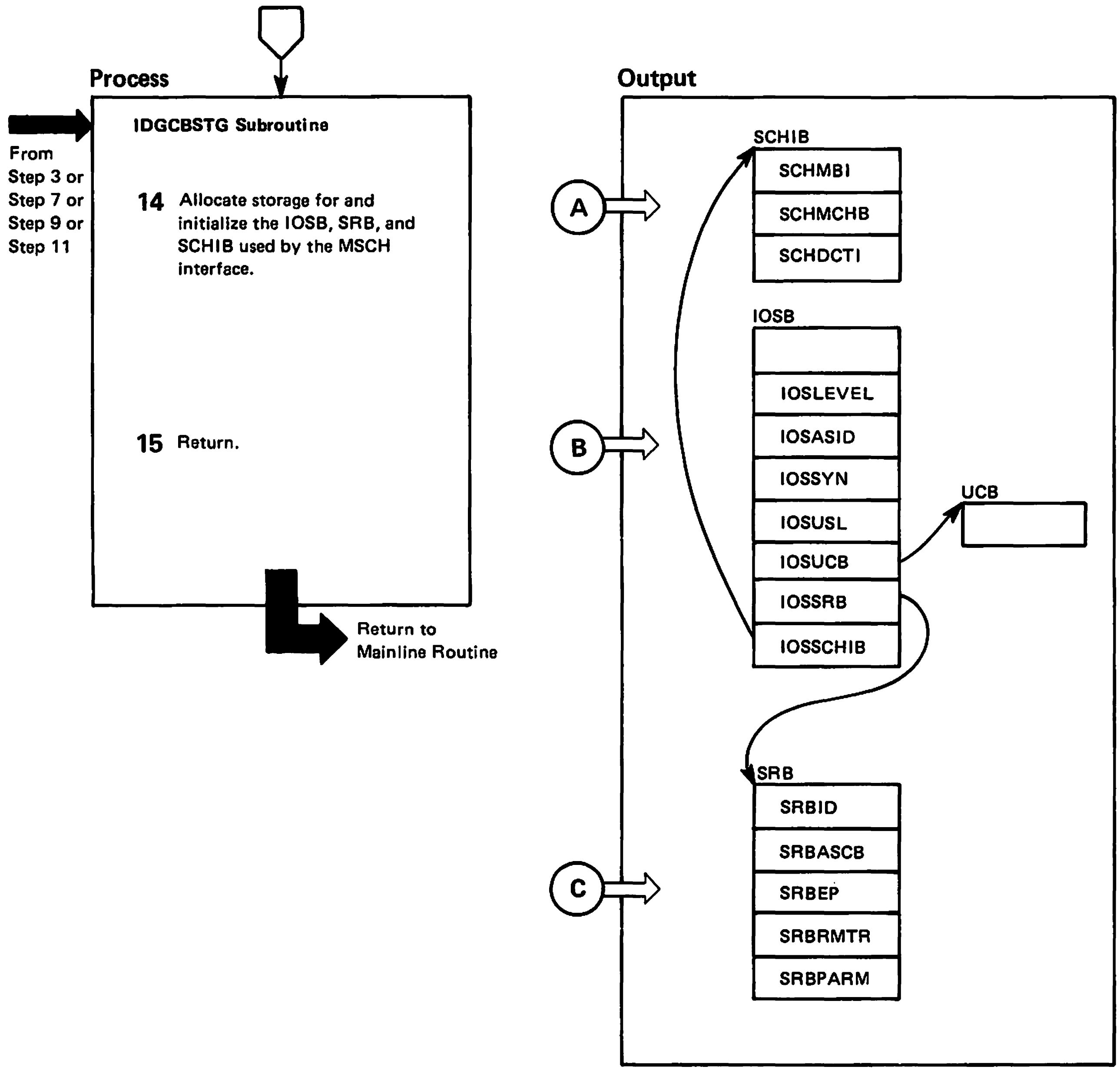


Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 10 of 12)

Extended Description	Module	Label
IDGCBSTG Subroutine		
14 Obtains storage for the IOSB, the SRB, and the SCHIB from subpool 245 and initializes these areas to binary zeroes. Initializes the appropriate fields in these control blocks for a MSCH request.		
15 Returns to the mainline routine.		

Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 11 of 12)

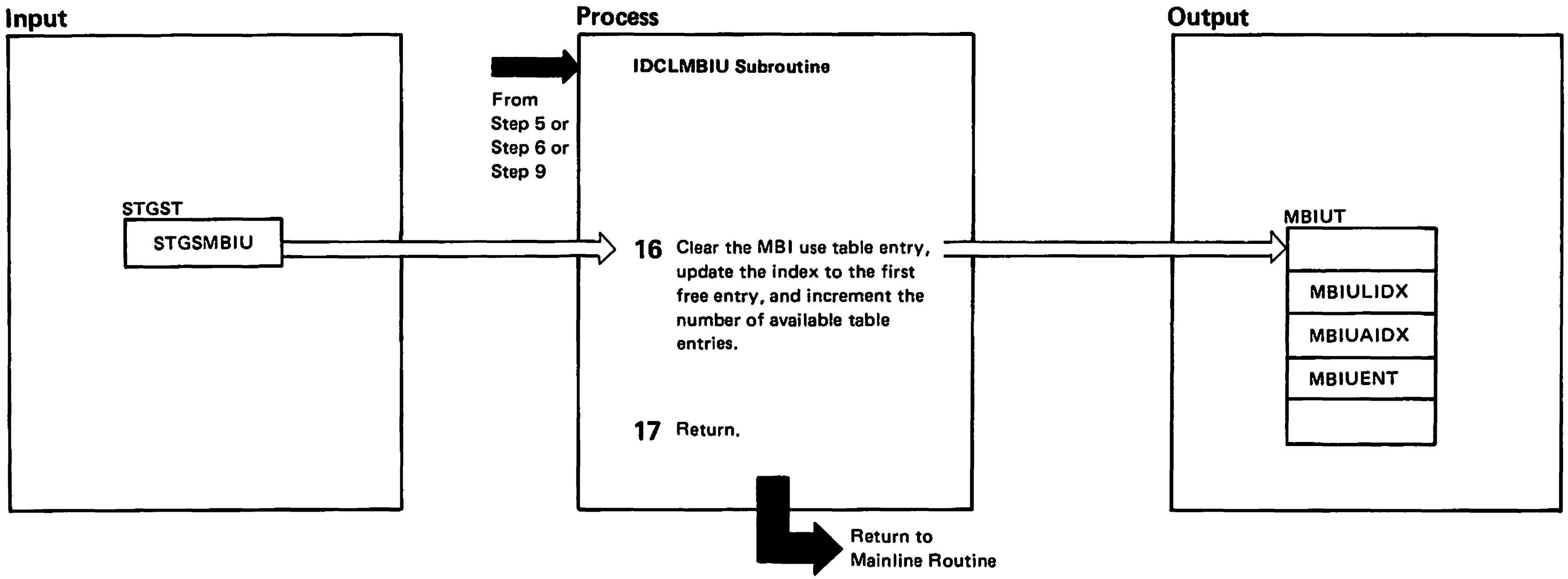


Diagram 28. Start/Stop Hardware Measurements for Devices (ERBMFIDA) (Part 12 of 12)

Extended Description	Module	Label

IDCLMBIU Subroutine

16 Clears the measurement block index (MBI) use table entry, updates the index of the first free entry, and increments the number of available table entries.

17 Returns to the mainline routine.

ERBMFIQA - MODULE DESCRIPTION

DESCRIPTIVE NAME: Start/Stop Hardware Measurements for I/O Queuing

FUNCTION:
Activates or deactivates the generation of I/O queuing hardware measurements.

ENTRY POINT: ERBMFIQA

PURPOSE: See function

LINKAGE: BALR

CALLERS:
```
   ERBMFIOQ - I/O queuing initialization,
   ERBMFTMA - Termination mainline
```

INPUT:
```
   Parameter 1 - IQREQCF: Operation request flag
   Parameter 2 - IQIOQTVP: Pointer to the I/O queuing
                 event data table (EIOQED)
```

OUTPUT:
Fields within the I/O queuing event data table (EIOQED) and the I/O queuing data blocks (EIOQDBs) are manipulated.

EXIT NORMAL: Returns via BR 14 to caller.

EXIT ERROR:
This module provides its own ESTAE exit routine, ERBIQERV. If an error occurs during processing when ERBIQERV is active, control returns to ERBMFIQA at retry label IQARETRY for cleanup processing.

EXTERNAL REFERENCES: See below

ROUTINES:
```
   ERBCNFGR - Configuration data retrieve module
   ERBMFMPR - Message processing module
   IOCVSTSQ - IOS/STSCH interface to get
              SCHIB data
```

CONTROL BLOCKS:
```
   CVT      - Communication vector table
   ERBSTGST - RMF global data
   ERBIOML  - I/O measurement level constants
   ERBIOPQ  - IOP initiative queue area
   ERBIOQDB - I/O queuing data block
   ERBIOQED - I/O queuing event data table
   ERBHSAD  - Hardware system area directory
   ERBHSARB - Hardware system area data
              request block
   ERBMFLMM - Language part index constants
   ERBMFMID - Message index table
   ERBPSCH  - Pseudo subchannel
   ERBSCHB  - Hardware system area subchannel block
   IECDIOCM - I/O communication area
   IHAPSA   - Prefixed save area
   IHASDWA  - System diagnostic work area
   IRACMCT  - Channel measurement control table
   IRARMCT  - SRM control table
```

ERBMFIQA - MODULE OPERATION

1. Establishes an ESTAE recovery environment.

2. Initializes for processing.

3. If RMF is running on a 308x processor:

   a. Sets the hardware measurement active or inactive flag in the I/O queuing data blocks.

4. If RMF is running on a 4381 processor:

   a. Obtains storage for and clears the hardware system area request block used by the SVC 122 interface.

   b. Depending on the request type parameter, sets up for activation or deactivation of hardware measurement generation and issues the SVC 122.

   c. If the activation/deactivation of hardware measurement generation fails, issues message ERB281I. If it succeds, sets the hardware measurement active or inactive flag in the I/O queuing data blocks.

5. If RMF is running on an IBM 3090 processor:

   a. Sets up channel measurement facility active/not active flag (start request only).

   b. Issues ERBDIAG macro instructions (DIAGNOSE interface) to get the address of the hardware system area (HSA) and to read the HSA-directory (start request only).

   c. Loops through the HSA-directory to get the address of the IOP initiative queue area and the address of the HSA-subchannel area, and issues the ERBDIAG macro instruction to read the IOP initiative queue area (start request only).

   d. For each selected LCU, issues the ERBDIAG macro instruction to read its associated HSA-subchannel area and saves the pointer to its CU-HDR (start request only).

   e. Depending on the request type parameter, sets up for activation or deactivation of hardware measurement generation and issues the ERBDIAG macro instruction.

   f. On successful completion, sets the hardware measurement active or inactive flag in the I/O queuing data blocks. If the activation request fails, retries with other target devices in the same LCU.

6. Frees any storage obtained for the hardware system area request block or hardware system area directory.

7. Deletes the ESTAE recovery environment and returns to the caller.

ERBMFIQA - MODULE OPERATION (Continued)

RECOVERY OPERATION:

Recovery is performed by the internal ESTAE exit routine ERBIQERV. If an error occurs during processing when ERBIQERV is active, control returns to ERBMFIQA at retry label IQARETRY for cleanup processing.

ERBMFIQA - DIAGNOSTIC AIDS

ENTRY POINT NAME: ERBMFIQA

MESSAGES:

ERB281I - UNABLE TO ACTIVATE/DEACTIVATE LCU MEASUREMENTS. RESPONSE CODE / RETURN CODE ccc. The activation or deactivation of hardware measurements failed. The message includes either the response code or the return code.

ABEND CODES:

1204 - Unexpected return code from configuration retrieve module, ERBCNFGR

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

- 0 - normal return, requested function completed normally
- 4 - requested function not completed normally

EXIT ERROR:

- 8 - the ESTAE error recovery exit terminated processing

REGISTER CONTENTS ON ENTRY:

Register 1 - Parameter list address
Register 13 - Save area address
Register 14 - Return address
Register 15 - Entry point address

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

Register 0 - 14 Restored to their original values
Register 15 - Return code

EXIT ERROR:

Register 0 - 14 Restored to their original values
Register 15 - Return code

ERBMFIQA - Start/Stop Hardware Measurements for I/O Queuing   STEP 01

ERBMFIOQ - I/O queuing initialization, ERBMFTMA - Termination mainline

PARAMETERS

IQREQFVC IQIOQTVP

ERBMFIQA

Activates or deactivates the generation of I/O queuing hardware measurements.

01 Performs initialization.

PARAMETERS

IQREQFVC

STGST

STGSIOQ

A. If termination processing was requested and I/O queuing was not active, returns to the caller.

STGST

STGSIOQ

PARAMETERS

IQIOQTVP

B. Sets up a pointer to the I/O queuing event data table.

C. Establishes an ESTAE recovery environment.

EIOQED

EIOQIOML

ERBIOML

IOML308X

02 If RMF is running on a 308x processor, performs required processing.

PARAMETERS

IQREQFVC

ERBIOQDB

EIOQDB

EIOQED

EIOQDBCT EIOQLCDB

A. Depending on the request type, sets the hardware measurements active or inactive flag in the I/O queuing data blocks.

ERBIOQDB

EIOQNHMA

EIOQED

EIOQIOML

ERBIOML

IOML4381

03 If RMF is running on a 4381 processor, performs required processing.

ERBHSARB

HSARB

A. Obtains storage for hardware system area request block (HSARB) and clears the obtained area.

IQPARMVC

IQHSAVP

ERBHSARB

HSARB

ERBMFIQA - Start/Stop Hardware Measurements for I/O Queuing — STEP 03B

IQCMDWVC

ERBHSARB

HSARB

B. Initializes the parameter list, command word, and header fields in the HSARB necessary for the SVC 122.

IQPARMVC

IQCMDWVP

IQCMDWVC

IQCCDEVB
IQCPARVB
IQCCLSVB

ERBHSARB

HSARLGTH

ERBHSARB

HSARRSPC HSARMDRC

C. Initializes the response code.

ERBHSARB

HSARRSPC
HSARMDRC

PARAMETERS

IQREQFVC

ERBHSARB

HSARPARM HSARCD
HSARFRMF HSARAM
HSARDM

D. Depending on the request type, sets up the request for the activation or deactivation of hardware measurement generation.

ERBHSARB

HSARFCD
HSARSFCD
HSARPARM
HSARCD

IQPARMVC

E. Issues SVC 122 within a loop. If SVC 122 issues a return code 4 or 8, indicating that the service processor is temporarily busy, retries the request after waiting one second.

ERBHSARB

HSARRSPC HSARNC20

F. If SVC 122 is unsuccessful, ERBMFIQA loads the message processing module (ERBMFMPR), builds and issues message ERB281I, deletes ERBMFMPR, sets a return code of 4 (to request termination of I/O queuing activity) and returns to the caller.

LOAD

EP=ERBMFMPR

PARAMETERS

IQREQFVC

STGST

STGSDCBA

ERBMFLMM

LMPART72 LMPRT148
LMPRT263 LMPRT264

ERBHSARB

HSARRSPC

ERBMFMPR

MID281I, IQVTXTVP(1),
IQNORPCF, IQDCBVP, IQSESICC

DELETE

EP=ERBMFMPR

ERBMFIQA - Start/Stop Hardware Measurements for I/O Queuing

STEP 03G

PARAMETERS
IQREQFVC
ERBIOQDB
EIOQDB
EIOQED
EIOQDBCT EIOQLCDB

G. Depending on the request type, sets the hardware measurements active or inactive flag in the I/O queuing data blocks.

ERBIOQDB
EIOQNHMA

EIOQED
EIOQIOML
ERBIOML
IOML3090
PARAMETERS
IQREQFVC

04 If RMF is running on an IBM 3090 processor, performs required processing.

IRACMCT
CMCTMFA

A. Sets up channel measurement available/not available flags in the I/O queuing event data table.

EIOQED
EIOQNCMF
EIOQACMF

ERBHSAD
HSAD

B. Obtains storage for hardware system area (HSA) directory and clears the obtained area.

C. Issues ERBDIAG macro instructions to get the address of the hardware system area (HSA) and to read the HSA-directory.

ERBHSAD
HSADENT HSADENAM
ERBHSAD
HSADEADR

D. Searches the HSA-directory to find the addresses of the pseudo subchannel, the IOP initiative queues, and the HSA subchannels. If it cannot find any address, ERBMFIQA sets the DIAGNOSE interface failure indicator (EIOQDIF) and skips the next step.

EIOQED
EIOQDIF

EIOQED
EIOQDIF
ERBPSCH
PSCHIOP0 PSCHIOP1
ERBIOPQ
IOPQID

E. If the previous step encountered an error, issues ERBDIAG macro instructions to read the pseudo subchannel and the IOP initiative queue area, then initializes the queue numbers.

ERBMFIQA - Start/Stop Hardware Measurements for I/O Queuing STEP 04F

EIOQED
EIOQDIF
ERBIOQDB
EIOQNUCB
ERBSCHB
SCHB
EIOQED
EIOQDBCT EIOQLCDB
ERBIOQDB
EIOQSCHN
ERBSCHB
SCHBCHDR

F. Loops through the I/O queuing data blocks and issues ERBDIAG for each LCU to get the pointer to the associated CU-HDR, then issues ERBDIAG to request activation of hardware measurement generation.

ERBIOQDB
EIOQNHMA
EIXQCUH

G. If the ERBDIAG completes successfully, bypasses retry steps.

>IQSUCACT: 04M

ERBIOQDB
EIOQNHMA
ERBIOQDB
EIOQDCNT

H. If the DIAGNOSE was unsuccessful, retries with other target devices from the same LCU.

ERBIOQDB
EIOQNHMA

ERBCNFGR
IQFCT3CC, EIOQLCUN, EIOQDNUM, IQNDVNVF

ERBIOQDB
EIOQDNUM

I. Issues IOSLOOK to get the pointer to the associated UCB and IOSINFO to get the subchannel ID.

ERBIOQDB
EIOQUCBA

J. Issues ERBDIAG to request the activation of hardware measurement generation.

ERBIOQDB
EIOQNHMA

K. If the request completed successfully, bypasses next step.

>IQSUCACT: 04M

L. Indicates that hardware measurement generation was not activated.

ERBIOQDB
EIOQNHMA

ERBIOQDB
EIOQDB

04M
IQSUCACT

M. Continues processing after successful activation.

ERBIOQDB: EIOQDB EIOQNHMA
EIOQED: EIOQDBCT EIOQLCDB

N. If termination processing was requested, loops through the I/O queuing data blocks and issues the ERBDIAG macro for each LCU to request deactivation of hardware measurement generation.

Output: ERBIOQDB: EIOQNHMA

IQARETRY

05 If the ESTAE exit was active, sets the return code to 8 to request termination of I/O queuing activity because of ESTAE processing.

06 Performs clean up processing.

A. Releases previously obtained storage.

IQPARMVC: IQHSAVP

FREEMAIN
(RU) A((IQHSAVP)) LV(IQHSALVF) SP(IQHSASVF)

B. Deletes the ESTAE recovery exit.

ESTAE
0, MF=(E,IQESTAPL)

C. Returns to the caller.

ERBIQERV

07 ERBIQERV - ESTAE error recovery routine.

SDWA: SDWACMPC SDWAEAS SDWACOMU

A. Provides module ID, date, and version, and sets up the environment for module ERBMFIQA.

B. To prevent recursion, indicates that the ESTAE recovery routine was active.

SDWA: SDWAMLVL SDWARRL

C. Sets up information in the SDWA.

Output: SDWA: SDWAMODN SDWACSCT SDWAREXN SDWACID SDWASC SDWAMLVL SDWARRL SDWACIDB

STGST: STGSIOCH STGSIODN STGSIOCT

D. Provides serviceability information in the variable recording area (IHAVRA).

E. Activates the DAE service.

ERBMFIQA - Start/Stop Hardware Measurements for I/O Queuing STEP 07F

STGST

STGSASID

F. Moves SDUMP (list form) from automatic storage and requests an SDUMP.

G. Returns to RTM.

SDWA

SDWASRSV

Diagram 30. Asynchronous MSCH Completion (ERBMFIDX) (Part 1 of 2)

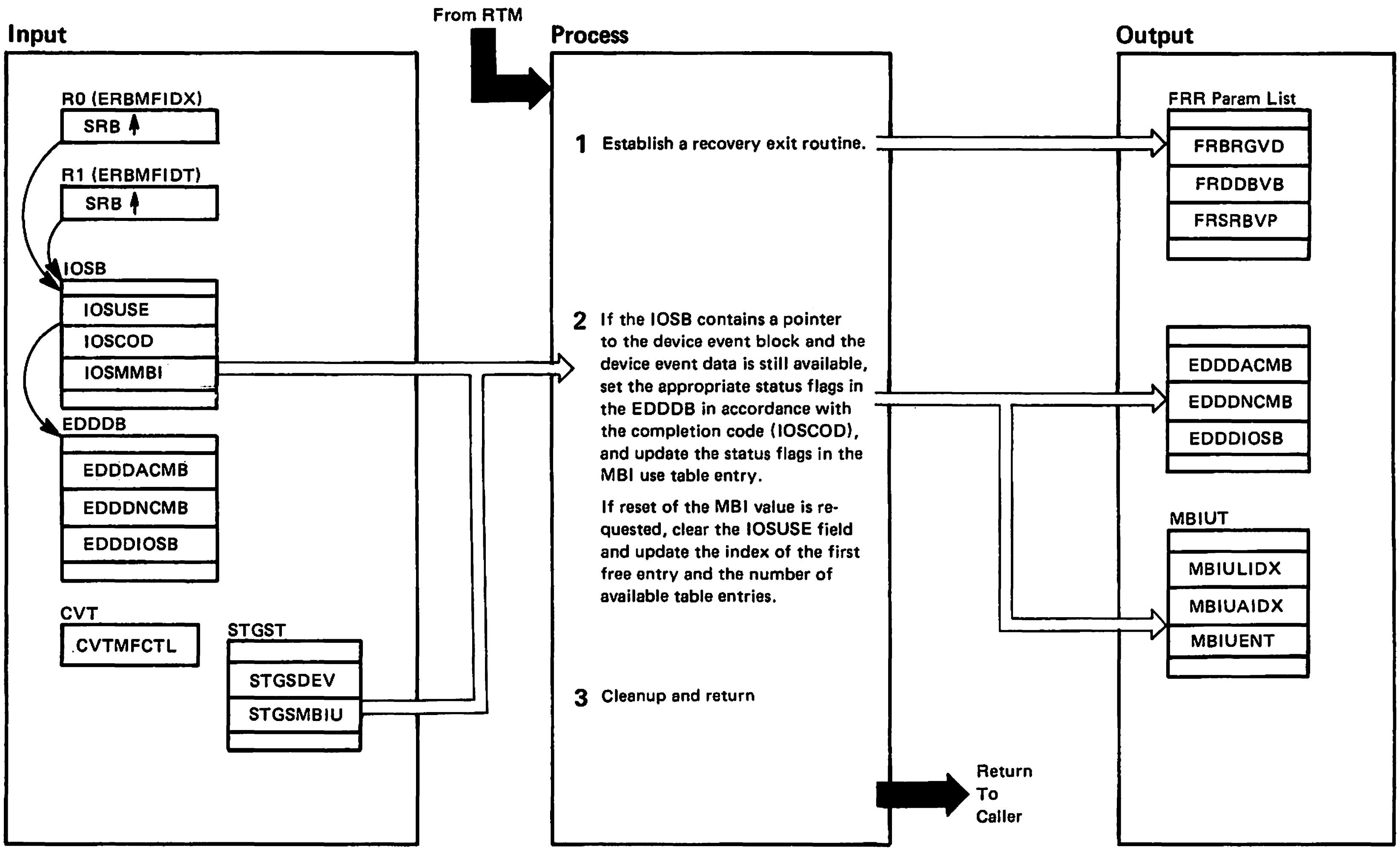


Diagram 30. Asynchronous MSCH Completion (ERBMFIDX) (Part 2 of 2)

Extended Description	Module	Label

ERBMFIDX is scheduled as an SRB routine on the completion of an asynchronous MSCH request issued by ERBMFIDA. It manipulates the hardware measurement available flags (EDDDACMB and EDDDNCMB) and/or performs cleanup operations. This module is scheduled on MSCH completion (entry ERBMFIDX) or receives control from RTM during PURGEDEQ request processing (entry ERBMFIDT). Processing for both calls is the same. The only difference is how the SRB address is passed. Once register 1 contains the SRB pointer, the logic is common for both entry points.

1 Builds a functional recovery routine (FRR) parameter list and issues the SETFRR macro instruction to establish a recovery routine (ERBMFIDF).

2 If the IOSUSE field in the IOSB contains a pointer (IOSMMBI = '1'B) to a device event data block (EDDDB) and device event data is still available, ERBMFIDX turns on (IOSCOD = X'7F') the CMB data now available flag (EDDDACMB); otherwise, ERBMFIDX turns this flag off and turns on the 'no CMB available' flag (EDDDNCMB), then updates the MBI use table to reflect the asynchronous completion and removes the IOSB address from the EDDDB to indicate that the asynchronous MSCH has completed.

If reset of the MBI value was requested (IOSMMBI = '0'B), the IOSUSE field contains the index of the associated MBI use table entry. In this case, ERBMFIDX clears the associated MBI use table entry; then eventually updates the index of the first free MBI use table entry and increments the number of available table entries.

3 ERBMFIDX frees the storage occupied by the IOSB, SRB, and SCHIB. It then cancels the FRR exit and returns to the caller.

Extended Description	Module	Label

Recovery Processing

ERBMFIDF establishes addressability from information provided in the FRR parameter list. If SDWAEAS=0, set up error information in the SDWA and issue the SDUMP macro instruction with SDATA=SQA, LSQA, TRT, PSA, SUMDUPM.

If the error in ERBMFIDX occurred during EDDDB update, free the storage occupied by the IOSB, SRB, and SCHIB.

Issue SETRP to request recording of the error and return to RTM.

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 1 of 14)

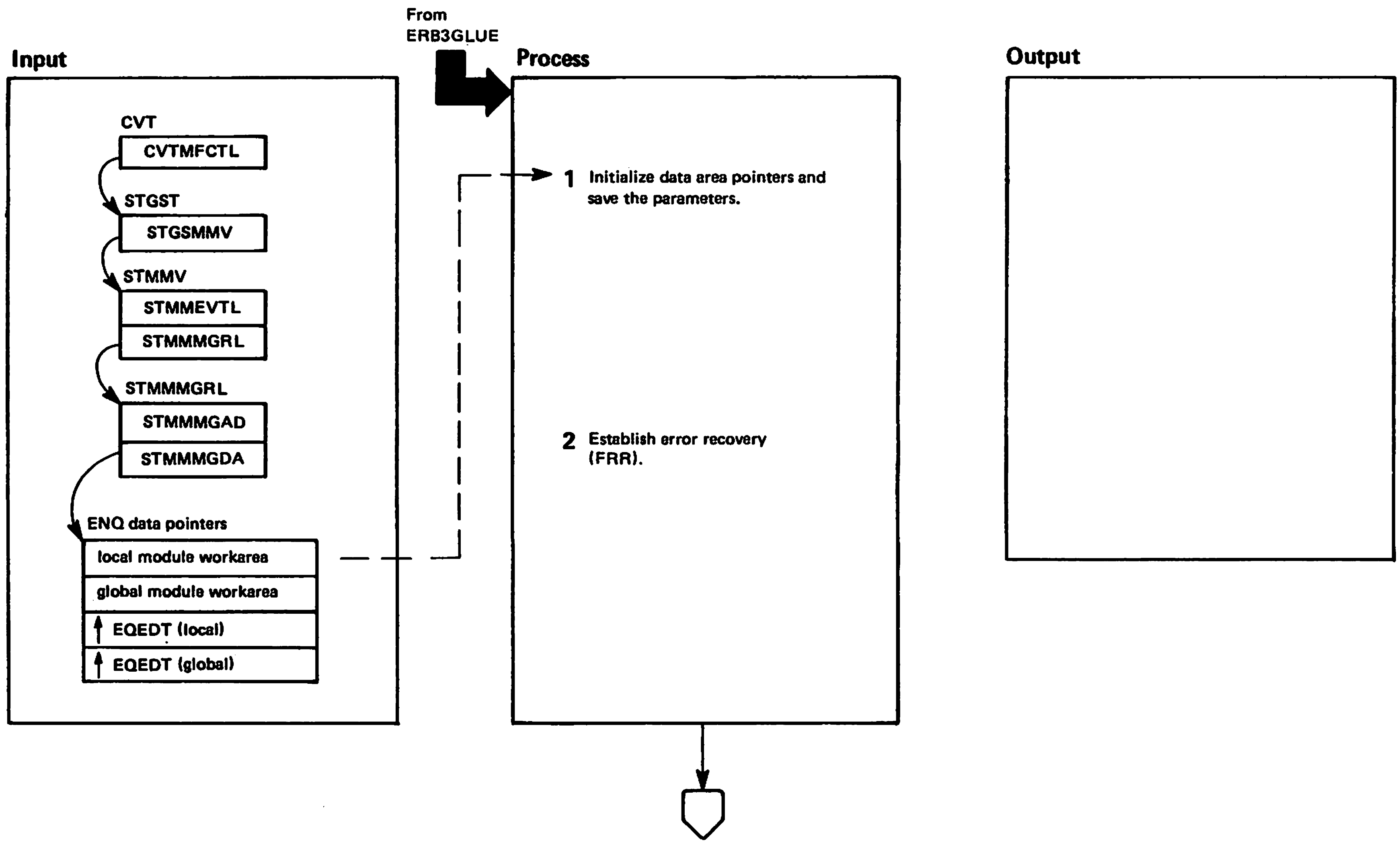


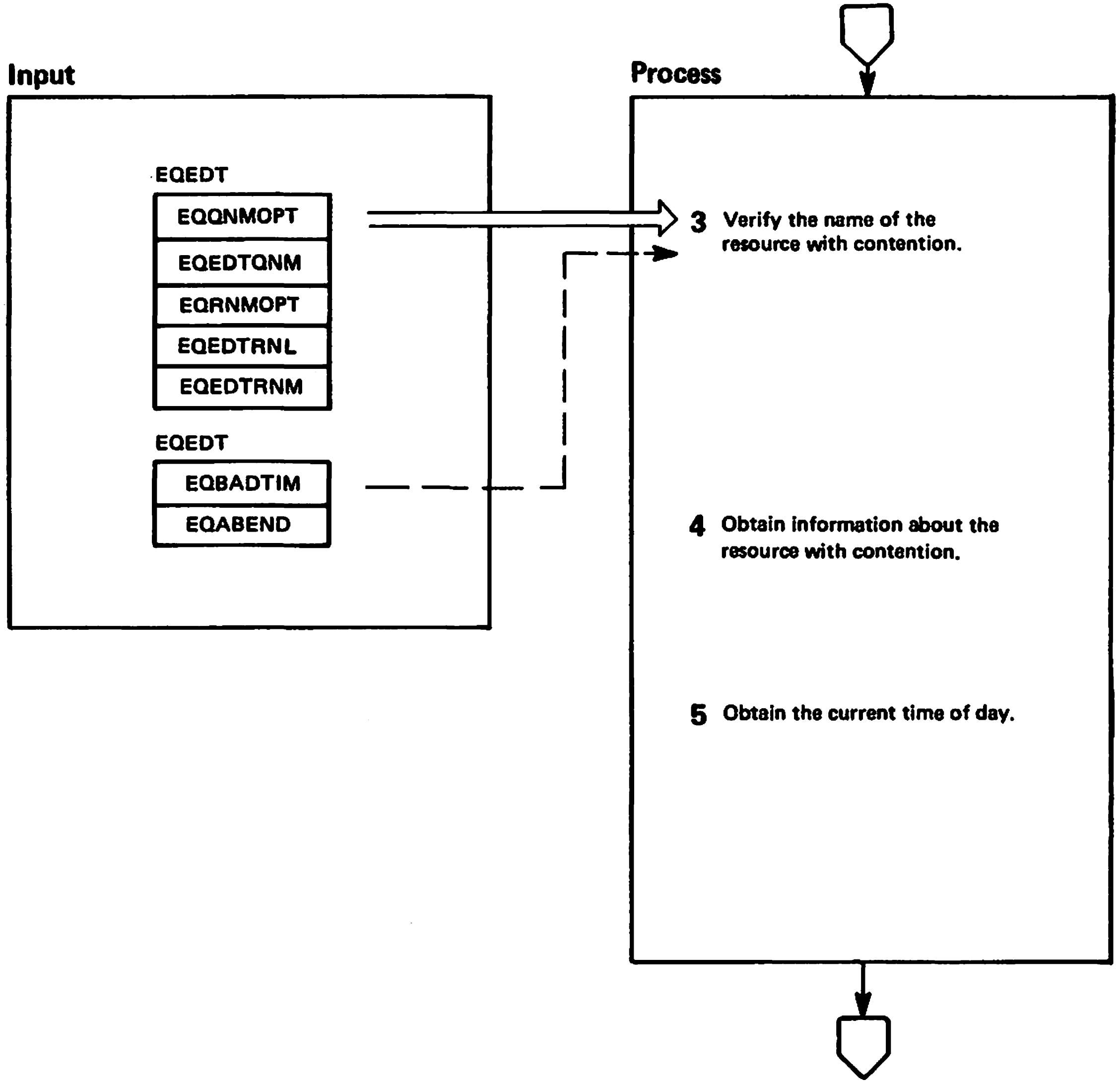
Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 2 of 14)

Extended Description	Module	Label
The ENQ Event Handler routine receives control from the ERB3GLUE module each time an ENQHOLD or ENQRLSE sysevent is issued.		
The input parameters include the name of the resource in contention, whether the resource is local or global, and whether an ENQHOLD or ENQRLSE is to be issued. Global resource serialization issues an ENQHOLD to inform the SRM that the user holds a resource that another user is waiting to use. It issues an ENQRLSE when the user releases the resource.		
The following example illustrates the "history" of a contention event. ERBMFEEQ is called (via ERB3GLUE) to record the contention each time a sysevent is issued.		

User	Request	Contention	Sysevent
A	ENQ SHR	None	None
B	ENQ SHR	None	None
C	ENQ EXCL	Contention event begins (user C is delayed)	ENQHOLD for user A ENQHOLD for user B
D	ENQ EXCL	Maximum contention (user C and user D are delayed)	None
A	DEQ	Contention reduced	ENQRLSE for user A
B	DEQ	Contention reduced	ENQRLSE for user B ENQHOLD for user C
C	DEQ	Contention event ends	ENQRLSE for user C

Extended Description	Module	Label
1 Call the SETUP subroutine to establish addressability to the correct automatic data area, according to the type of resource (local or global). The SETUP subroutine stores the parameters passed by the caller in the automatic area.		SETUP
2 Issue the SETFRR macro instruction to establish the ERBMFFRQ routine as the error recovery routine, for ERBMFEEQ.		

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 3 of 14)



Extended Description	Module	Label
3 If the user specified a resource name, verify that the resource with contention has the same name as the user-specified resource. If it does not, go to step 11 to return to the caller. If there was an abend or the clock was bad on a previous call, go to step 11 to return to the caller.		
4 Issue the GQSCAN macro instruction, specifying the name of the resource with contention. GQSCAN returns data about the resource in the form of a resource information block (RIB) and one or more resource information block extents (RIBEs). If requests for the same resource have been made with different scopes, GQSCAN returns a RIB (and its associated RIBEs) for each scope.		
5 Obtain the current time of day for later use in computing the length of the contention event. For ENQHOLD, the time is the moment when a user's execution is delayed because the request is for a resource being held by another user. For ENQRLSE, the time is the moment when a contention situation has disappeared because the release of a resource by a user for whom an ENQHOLD had previously been received. If the current clock is bad, go to step 11 to return to the caller.		

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 5 of 14)

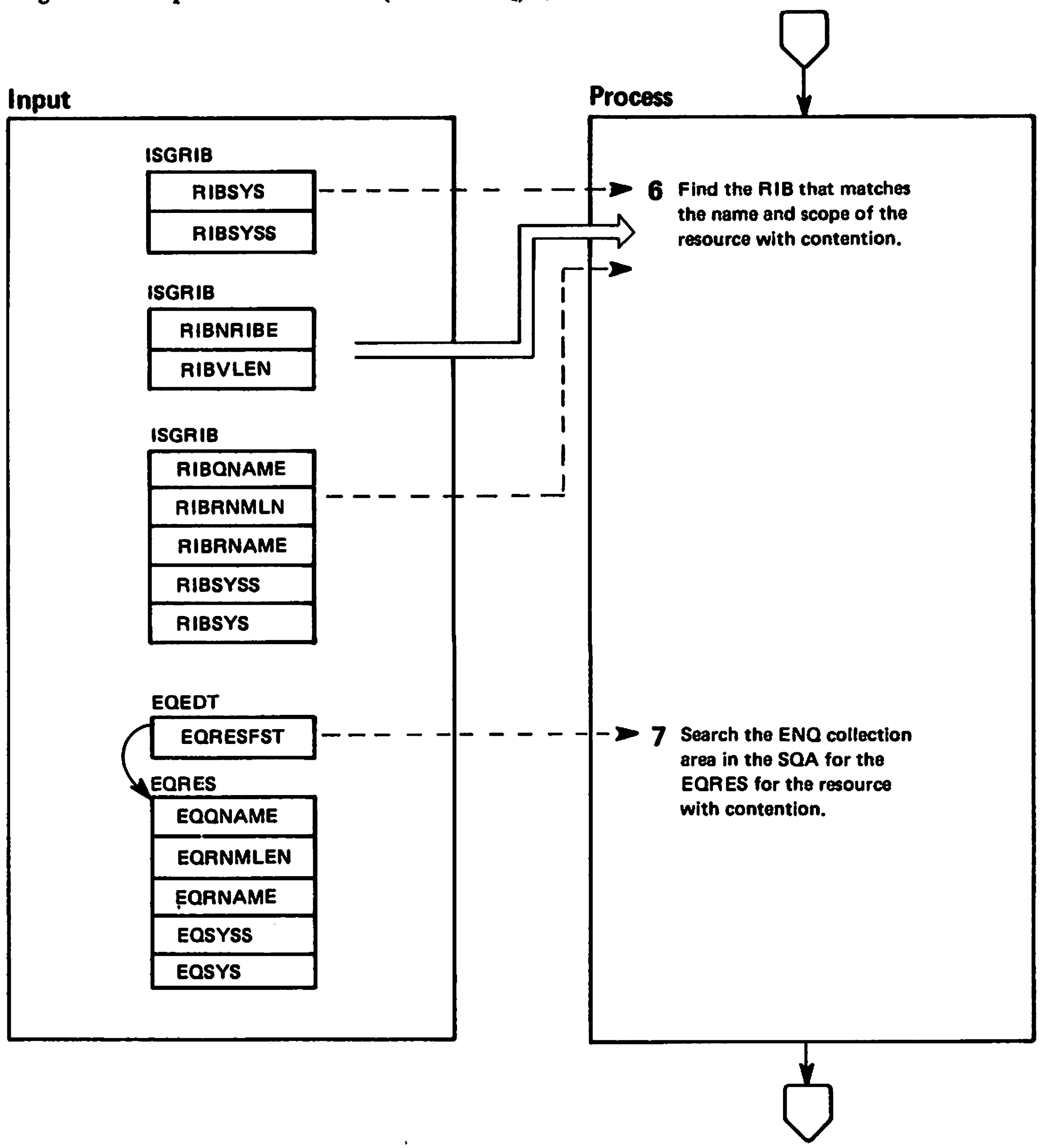


Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 6 of 14)

Extended Description	Module	Label
6 Search throught the RIBs for the resource under contention to locate the RIB that has the same scope as the desired resource. If a match is not found, issue GQSCAN again to get the rest of the data.		
7 Find the EQRES for the resource with contention. The EQRES is the control block that contains, according to name (QNAME and RNAME), information about contention for a serialized resource. If no EQRES exists for the resource, go to step 10 to build one.		

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 7 of 14)

Input

EQRES
- EQHLDT

EQRES
- EQWAITCNT

ISGRIB
- RIBNTWE
- RIBNTWS

ISGRIB
- RIBNTWE
- RIBNTWS

EQRES
- EQWAITCT

Process

8 If the EQRES for the input resource is found and ENQHOLD is issued:

a. If this is the start of a contention event, save the start data.

b. Set the hold count.

c. If the wait queue length is longer than before or unchanged, save the names of the owners and waiter(s).

GETCONT Step 13

9 If the EQRES for the input resource is found and ENQRLSE is issued, decrease the hold count. If the contention event has ended:

a. If the wait queue is longer than the current maximum, save the names of the owners and waiters.

GETCONT Step 13

Output

EQRES
- EQTOTEVN
- EQHLDT
- EQHLDCNT

EQRES
- EQHLDCNT

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 8 of 14)

Extended Description	Module	Label
8 When an EQRES exists for the resource and the sysevent is ENQHOLD, update the EQRES to reflect the additional contention:		
a. At the beginning of a contention event, record the start time.		
b. Increase the hold count (EQHLDCNT) by one.		
c. If the number of waiters in the RIB is greater than (or the same as) the wait count in EQRES, call the GETCONT subroutine to save the names of the current owners and waiters from the data in the RIBEs.		GETCONT
Go to step 11 to return to the caller.		
9 When an EQRES exists for the resource and the sysevent is ENQRLSE, update the EQRES to reflect the reduced contention. Decrease the hold count (EQHLDCNT) by one. If the hold count is zero, then the contention event has ended. If the contention event has ended, do the following:		
a. If the number of waiters in the RIB is greater than the count at any other time during this event, call the GETCONT subroutine to save the names from the data in the RIBEs.		GETCONT

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 9 of 14)

Input

EQRES

- EQHLDCNT
- EQHLDT
- EQWAITCT
- EQMAXHLD
- EQOWNERE
- EQWAIT1E
- EQWAIT2E
- EQOWNCTM
- EQWATCTM
- EQMINHLD
- EQTOTHLD
- EQQLEN1
- EQQLEN2
- EQQLEN3
- EQQLEN4
- EQTOTWQ
- EQOWNCNT
- EQOWNER1
- EQOWNER2
- EQWAITR1
- EQWAITR2

Process

9 (continued)

b. Update the maximum contention data.

Output

EQRES

- EQHLDCNT
- EQHLDT
- EQTOTHLD
- EQQLEN1
- EQQLEN2
- EQQLEN3
- EQQLEN4
- EQTOTWQ
- EQMAXHLD
- EQOWNCTM
- EQOWNCNT
- EQWATCTM
- EQWAITCT
- EQOWNEM
- EQWAT1EM
- EQWAT2EM
- EQOWN1M
- EQOWN2M
- EQWAT1M
- EQWAT2M
- EQMINHLD

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 10 of 14)

Extended Description	Module	Label
9 (continued)		
b. Update the total contention time and the queue length distribution. If it was longer than the previous contention events, update the maximum contention data. That is, move the data saved by GETCONT for this event into another part of the EQRES. This is the data that appears in the report under the heading JOBS AT MAXIMUM CONTENTION.		GETCONT

Go to step 11 to return to the caller.

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 11 of 14)

Input

EQEDT
- EQNXTTAB
- EQENDTAB
- EQRESFST

ISGRIB
- RIBRNMLN
- RIBSYSS
- RIBSYS

ISGRIB
- RIBRNMLN
- RIBQNAME
- RIBRNAME

Process

10 If an EQRES for the input resource is not found, build one.

Obtain data on owners and waiters.

GETCONT
Step 13

11 Delete the recovery environment.

12 Return to the caller.

To ERB3GLUE

Output

EQEDT
- EQNXTTAB
- EQDATAFL
- EQRESFST

EQRES
- EQRESNXT
- EQHLDCNT
- EQTOTEVN
- EQHLDT
- EQMINHLD
- EQRNMLEN
- EQQNAME
- EQRNAME
- EQSYSS
- EQSYS
- EQMINEXL
- EQMINSHR

Extended Description	Module	Label
10 No EQRES exists for the resource; this contention is the first for the resource during the reporting interval. Use the EQEDT to locate the next available EQRES. Build the EQRES from the data in the RIB.		
Call the GETCONT subroutine to save the names of the current owners and waiters from the data in the RIBEs.		GETCONT
11 Delete the error recovery environment (FRR) for ERBMFEEQ and reset the in-use bit in the STGST.	ERBMFEEQ	EEQTERM
12 Return to the caller.		

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 13 of 14)

Input

ISGRIB
- RIBNTWE
- RIBNTWS
- RIBNTO
- RIBNRIBE
- RIBESTAT
- RIBETYPE
- RIBVLEN
- RIBEJBNM
- RIBEASID
- RIBESYSN

EQRES
- EQMAXEXL
- EQMINEXL
- EQMAXSHR
- EQMINSHR
- EQWAITCT

Process

From Step 8C, Step 9A, or Step 10

GETCONT:

13 Place data about owners and waiters in the EQKES.

14 Return to caller.

To Step 8C, Step 9A, or Step 10

Output

EQRES
- EQWAITCT
- EQOWNCNT
- EQMAXEXL
- EQMINEXL
- EQMAXSHR
- EQMINSHR
- EQWAIT1E
- EQWAIT2E
- EQOWNERE
- EQOWNER1
- EQOWNID1
- EQOWN1SY
- EQOWNER2
- EQOWNID2
- EQOWN2SY
- EQWAITR1
- EQWATID1
- EQWAT1SY
- EQWAITR2
- EQWATID2
- EQWAT2SY

Diagram 31. Enqueue Event Handler (ERBMFEEQ) (Part 14 of 14)

Extended Description	Module	Label
13 When the wait queue is the longest within a contention event, "maximum contention," the mainline calls the GETCONT subroutine. GETCONT stores data about the first two owners and the first two waiters in the EQRES. It also saves the counts of all owners and waiters. If necessary, it issues GQSCAN again to obtain all of the RIBEs.	ERBMFEEQ	GETCONT
14 Return to the caller.		

Error Recovery

ERBMFRRQ, the error recovery routine for ERBMFEEQ, moves serviceability data to the VRA in the system diagnostic work area (SDWA). (See Section 6 — SDWA Variable Recording Area (VRA) for a description of the serviceability data in the variable recording area.) ERBMFRRQ also requests an SDUMP unless one was produced for a prior ABEND. It request a retry at step 11 to return control to the caller of ERBMFEEQ.

Diagram 32. RMF Message Processor (ERBMFMPR) (Part 1 of 2)

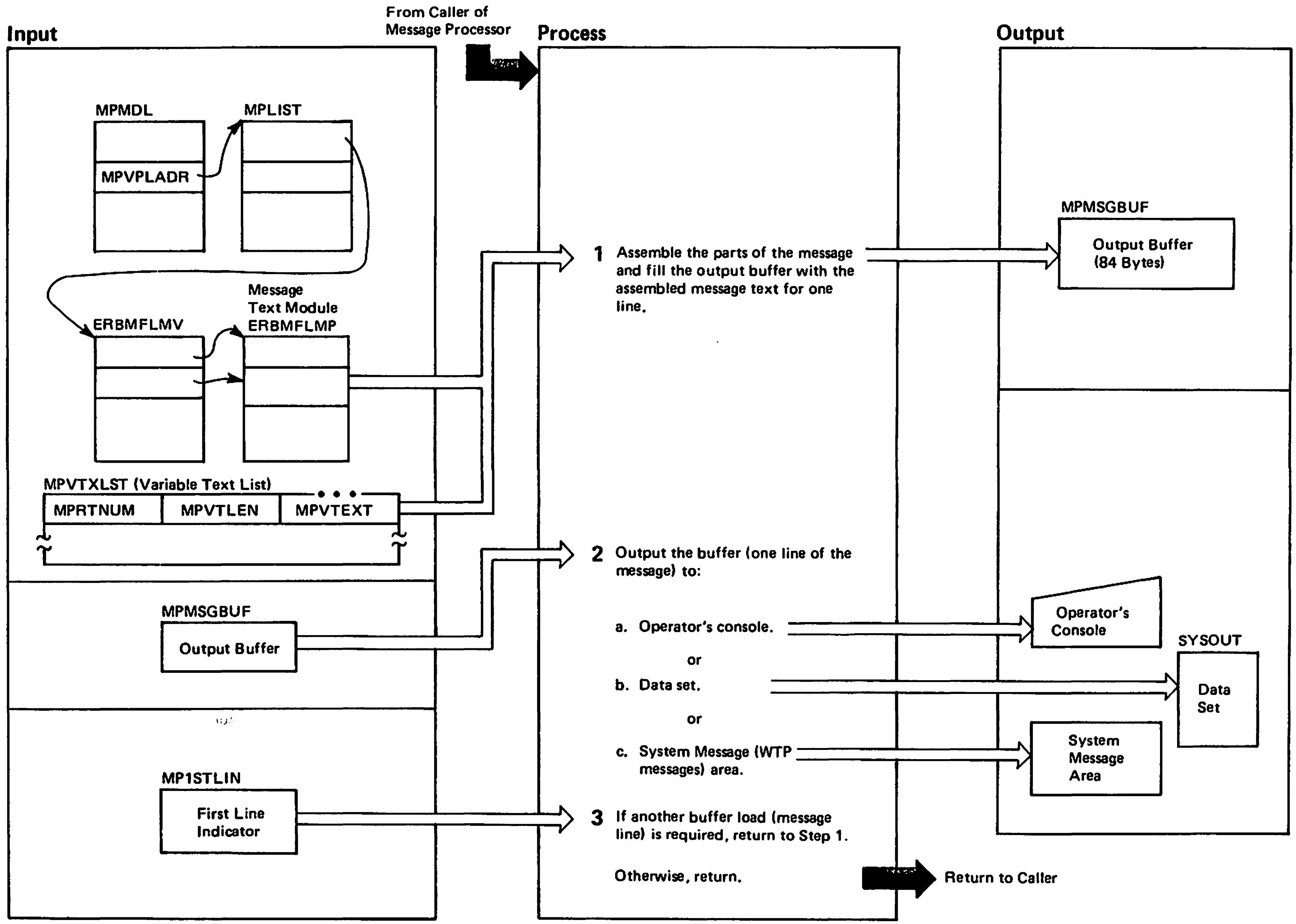
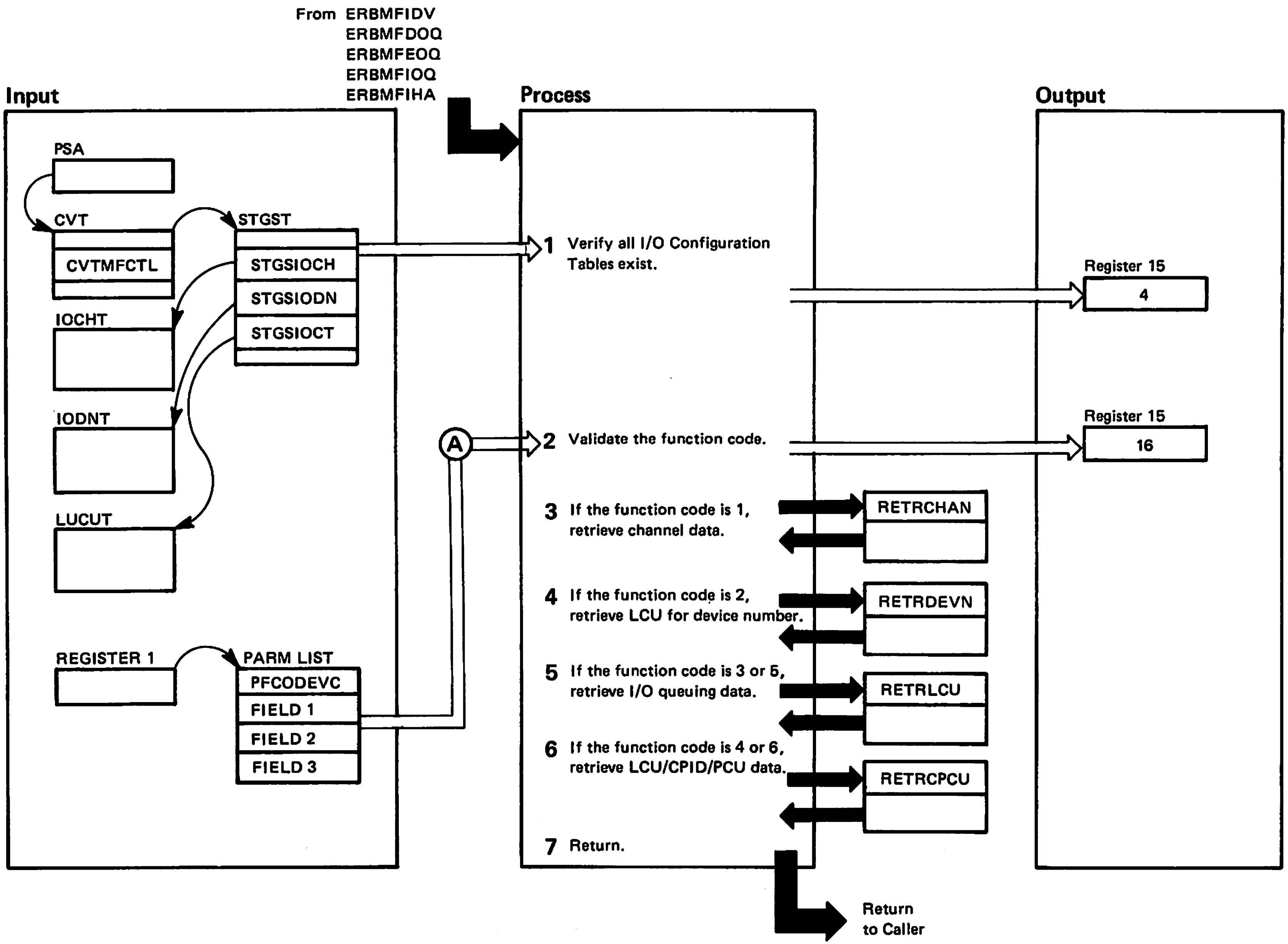


Diagram 32. RMF Message Processor (ERBMFMPR) (Part 2 of 2)

Extended Description	Module	Label
The Message Processor (ERBMFMPR) is called from several places in the RMF program to print output messages. (These are: ERBMFDTA, ERBMFINP, ERBMFRGM, ERBCNFGC, ERBMFPDU, ERBMFXCB, ERBMFPER, ERBMFMFC, and ERBMFMLN.) The Message Processor assembles the required message from parts in the Message Text module (ERBMFLMP), moves the parts into an output buffer, one message line at a time, and writes the message lines to the required output device or data set.	ERBMFMPR	
1 Input parameters define the message in terms of fixed and/or variable text portions. Fixed text portions are obtained from ERBMFLMP through an index in table MPLIST. When an MPLIST entry contains a zero, a variable text entry is obtained from the variable text list (MPVTXLST). If the variable text length (MPVTLEN) is non-zero, the variable text is moved into the buffer. If the variable text length is zero and the MPRTNUM field is non-zero, the MPRTNUM value is used to index into ERBMFLMV, to obtain fixed text from ERBMFLMP. Up to 80 bytes of message text and message identifier are assembled in the buffer.	ERBMFMPR	MFBLDMSG
2 The message Processor calls routine MFOUTMSG to write the buffer to the operator's console or required data set and then returns to the Message Processor as soon as the message is sent.	ERBMFMPR	MFOUTMSG
3 The message Processor controls the assembling of message lines and writing them until the entire message is sent.	ERBMFMPR	

Diagram 33. I/O Configuration Table Retrieve (ERBCNFGR) (Part 1 of 10)



Extended Description	Module	Label

ERBCNFGR retrieves data from the following I/O Configuration Tables:

- Channel path table (IOCHT)
- Device number table (IODNT)
- Logical control unit table (LCUT)

Depending upon a given function code (FC) as follows:

- FC=1 Retrieve from the IOCHT the channel path entry for a given CPID.
- FC=2 Retrieve from the IODNT the LCU associated with a given device number.
- FC=3 Retrieve from the IODNT the next device number associated with a given LCU.
- FC=4 Retrieve from the LCUT the total number of LCUs in the system.
- FC=5 Retrieve from IODNT the first device number associated with a given LCU.
- FC=6 Retrieve from LCUT the complete entry containing CPID and PCU information for a given LCU.

1 Validate that all the pointers (STGSIOCH, STGSIODN, STGSIOCT) to the I/O configuration tables in the STGST are non-zero. If true, the tables (IOCHT, IODNT, LCUT) exist. Otherwise, set a return code of 4 and return to the caller via Step 7.

2 Validate the function code (passed in the first entry of the parameter list). If it is not a numeric value between 1 and 6, set a return code of 16 and return to the caller via Step 7.

3 If the function code is 1, call the RETRCHAN subroutine to retrieve channel path data from the IOCHT.

4 If the function code is 2, call the RETRDEVN subroutine to retrieve LCU data for a given device number from the IODNT.

Extended Description	Module	Label

5 If the function code is 3 or 5, call the RETRLCU subroutine to retrieve device number data for a given LCU from the IODNT.

6 If the function code is 4 or 6, call the RETRCPCU subroutine to retrieve the total number of LCUs as well as channel path and physical control unit information from the LCUT.

7 Return to the caller with one of the following codes in register 15:

00 The information requested was retrieved and the data is available in the answer field provided by the caller.

04 The I/O configuration tables do not exist.

08 The search item was not valid.

12 The search item was valid, but an entry in the respective table was not found.

16 The function code was invalid.

Diagram 33. I/O Configuration Table Retrieve (ERBCNFGR) (Part 3 of 10)

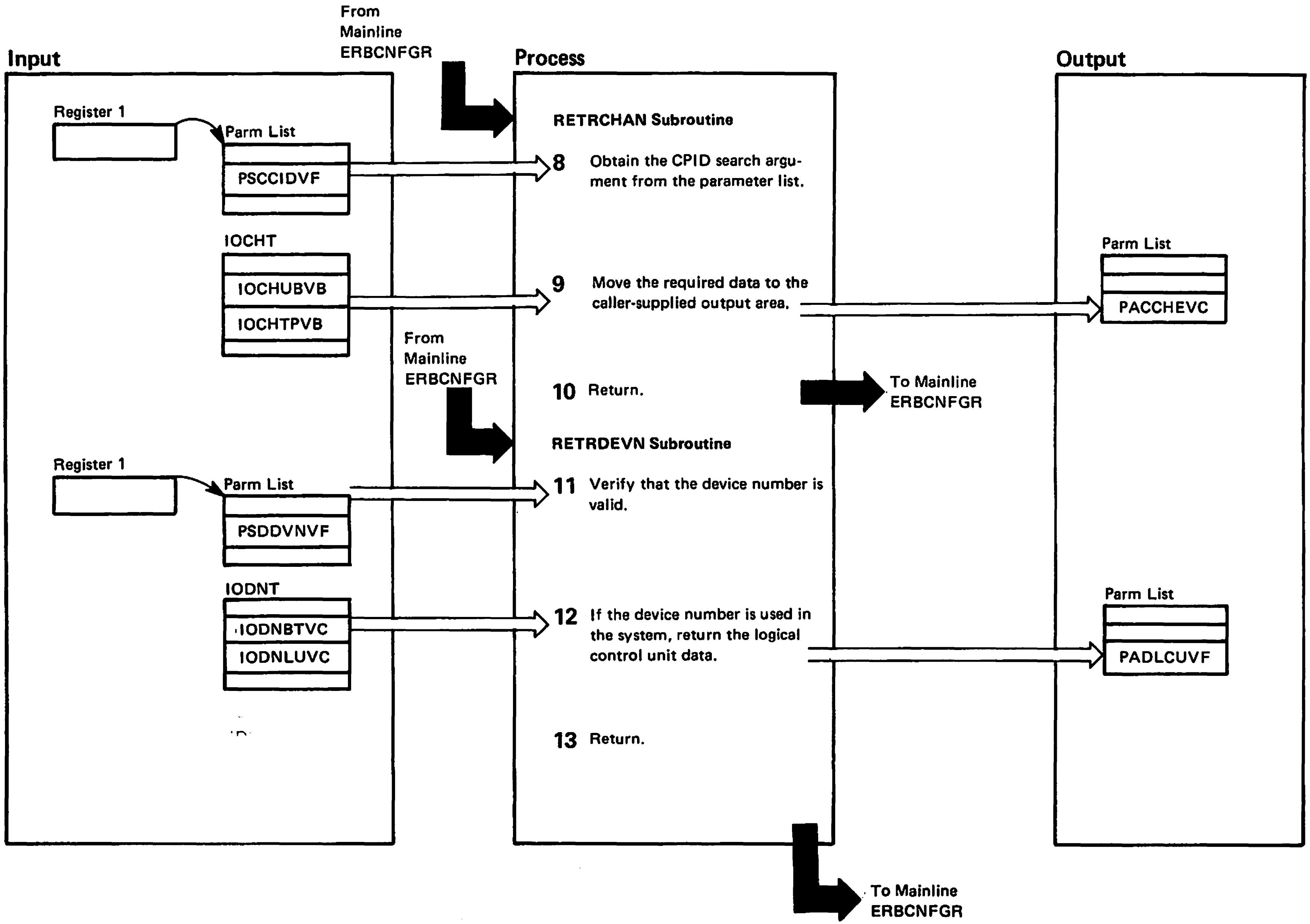


Diagram 33. I/O Configuration Table Retrieve (ERBCNFGR) (Part 4 of 10)

Extended Description	Module	Label

RETRCHAN Subroutine

8 The second entry in the parameter list contains a search argument (CPID).

9 The search argument (CPID) is used to locate the appropriate IOCHT entry and, if the channel path is installed, the complete IOCHT entry is moved to the area supplied as the third entry in the parameter list.

10 Return to the mainline with one of the following codes:

RC=00 The information requested was retrieved and the data is available in the answer field provided by the caller.

RC=12 The search item was valid, but an entry in the respective table was not found.

RETRDEVN Subroutine

11 Validate the device number contained in the second entry in the parameter list. The device number must be less than or equal to X'FFF'. For an invalid device number set a return code 8 and go to Step 13.

12 Use the device number to locate the appropriate entry in the IODNT. If the entry indicates that the device number is used by the system, return the associated LCU. Otherwise, set a return code of 12.

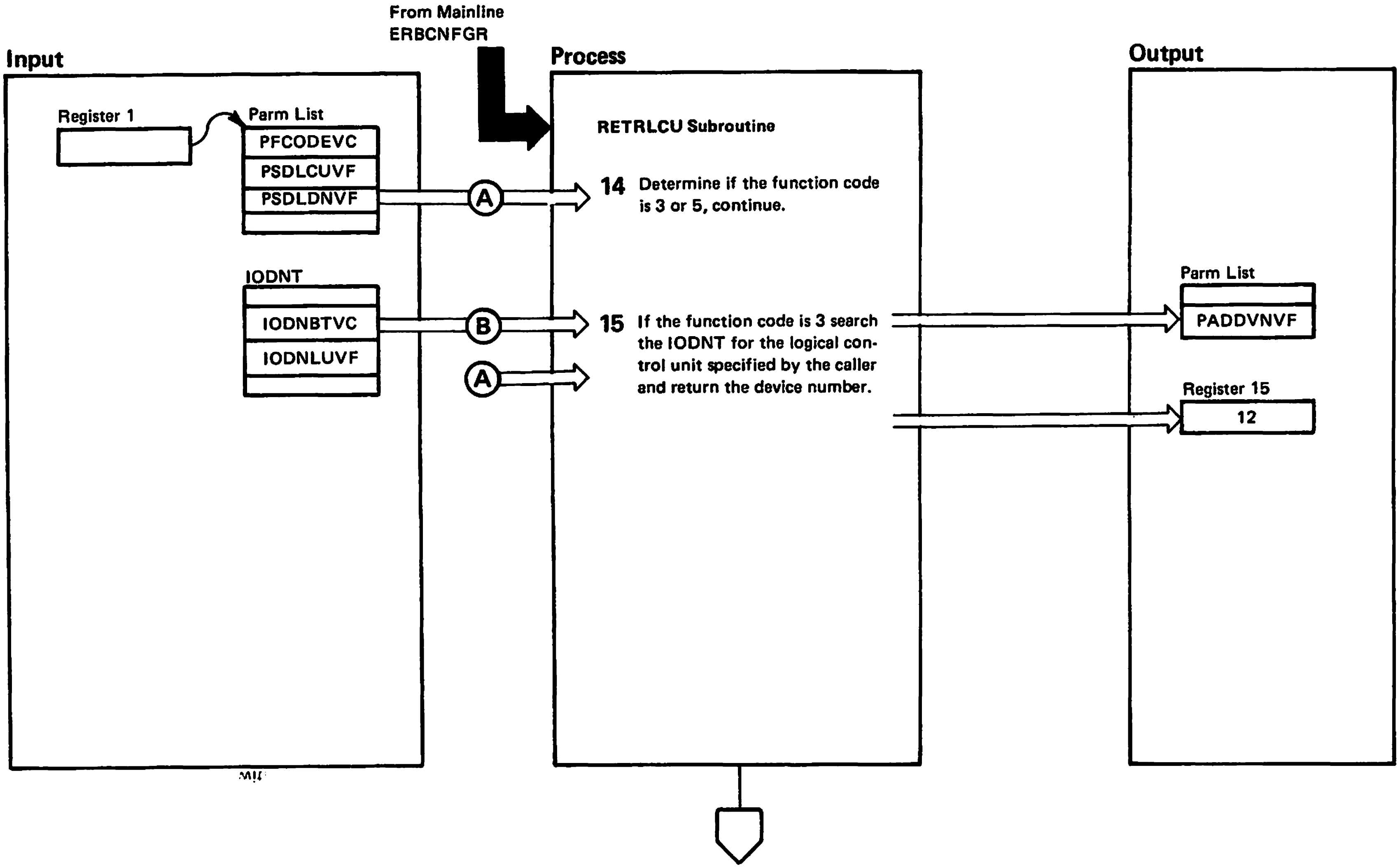
13 Return to the mainline with one of the following codes:

RC=00 The information requested was retrieved and the data is available in the answer field provided by the caller.

RC=08 The device number was invalid.

RC=12 The device number was valid, but an entry in the IODNT was not found.

Diagram 33. I/O Configuration Table Retrieve (ERBCNFGR) (Part 5 of 10)



Extended Description	Module	Label

RETRLCU Subroutine

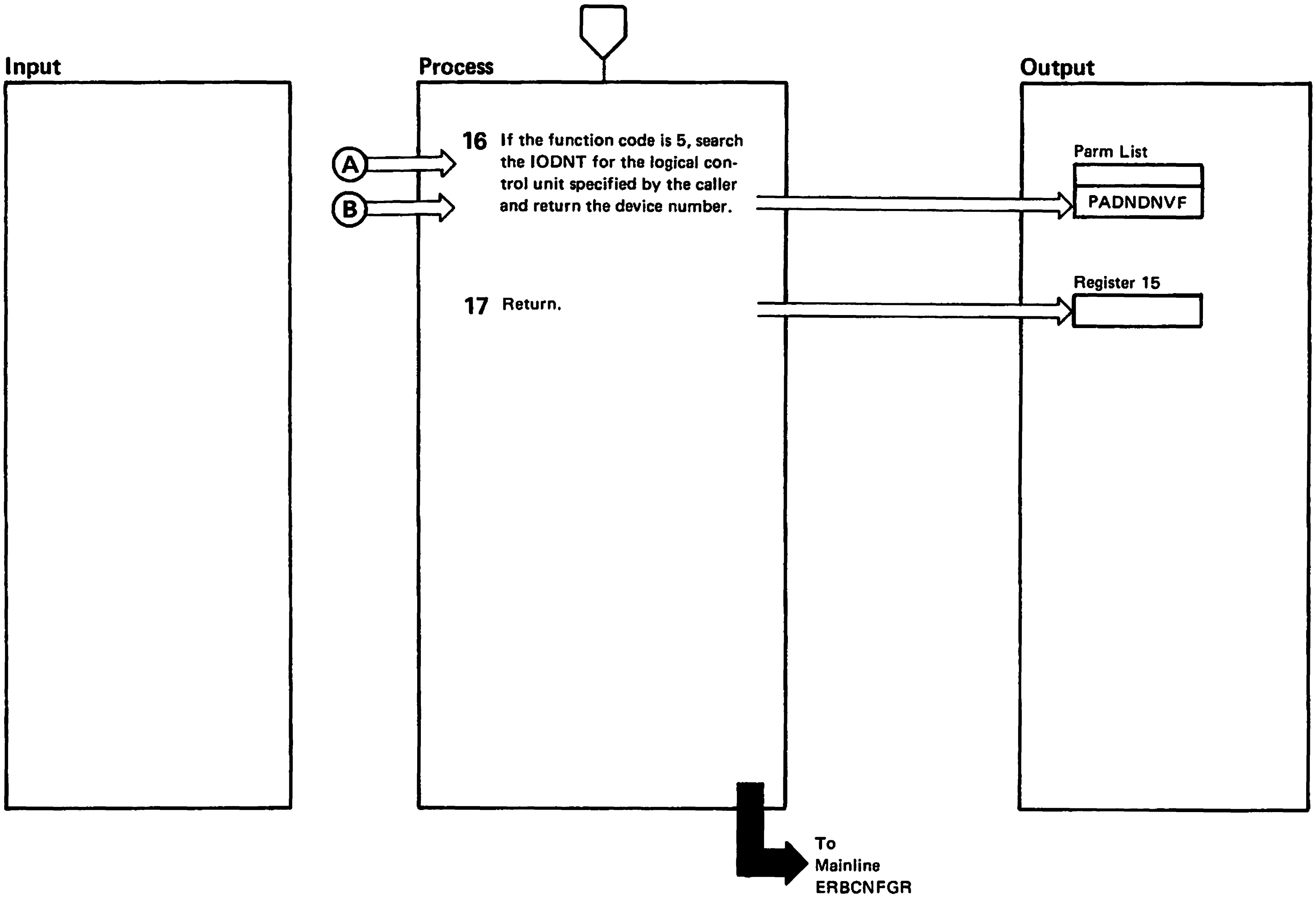
14 Depending on the function code, RETRLCU performs one of two functions:

- FC=5 (search on logical control unit) returns the first device number associated with the LCU specified in PSDLCUVF.
- FC=3 (search on logical control unit from device number) returns the next device number associated with the LCU specified in PSDLCUVF starting with the IODNT entry associated with the device number specified in PSDLDNVF plus one.

15 For a function code of 5 (retrieve first device number for a given LCU), loop through all IODNT entries. If the LCU from caller matches the LCU in the IODNT, then return the associated device number in the answer field and set a return code of 0.

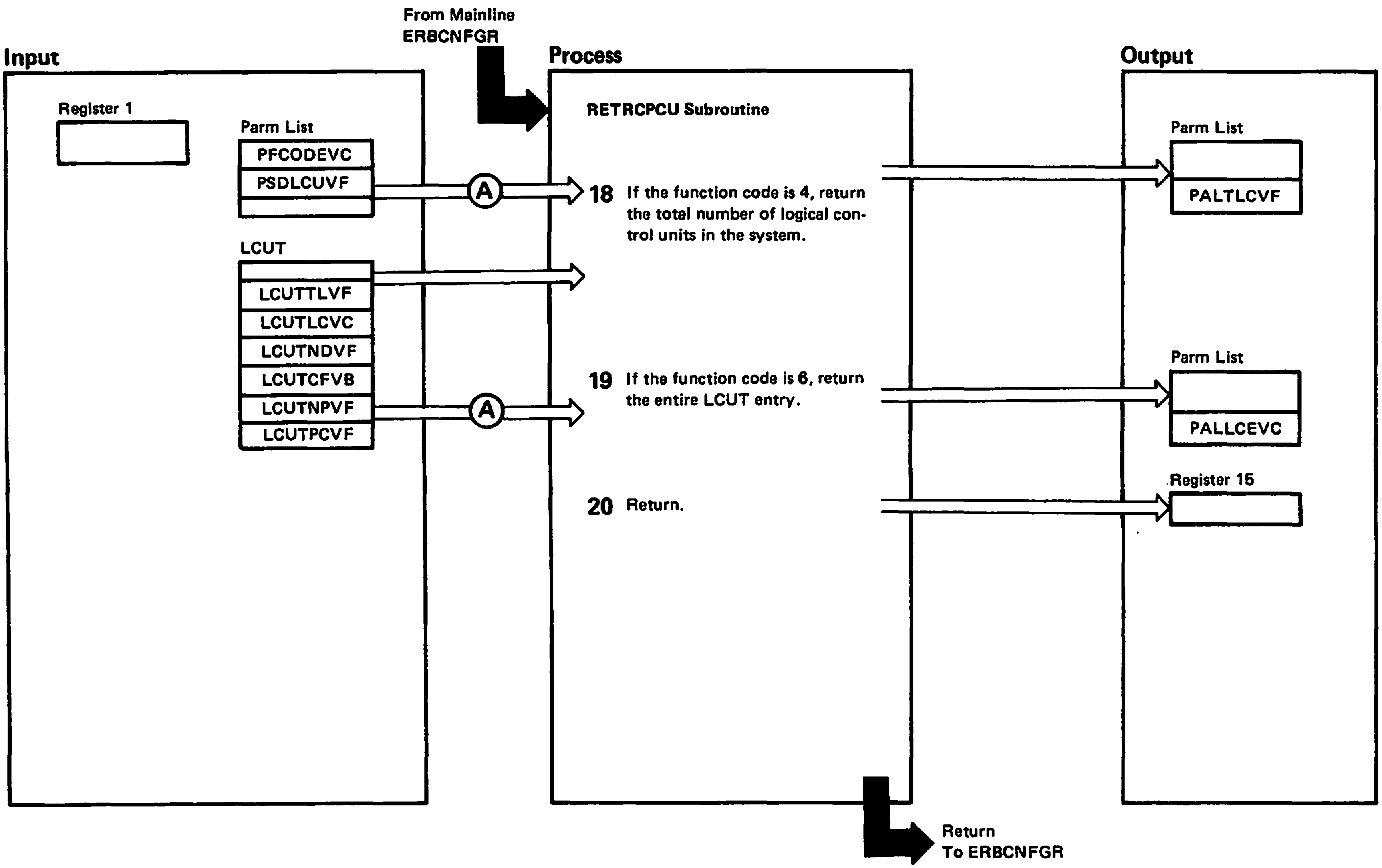
If end of IODNT is reached and no match occurred, pass a return code of 12 to the caller. In both cases, return to the caller via Step 17.

Diagram 33. I/O Configuration Table Retrieve (ERBCNFGR) (Part 7 of 10)



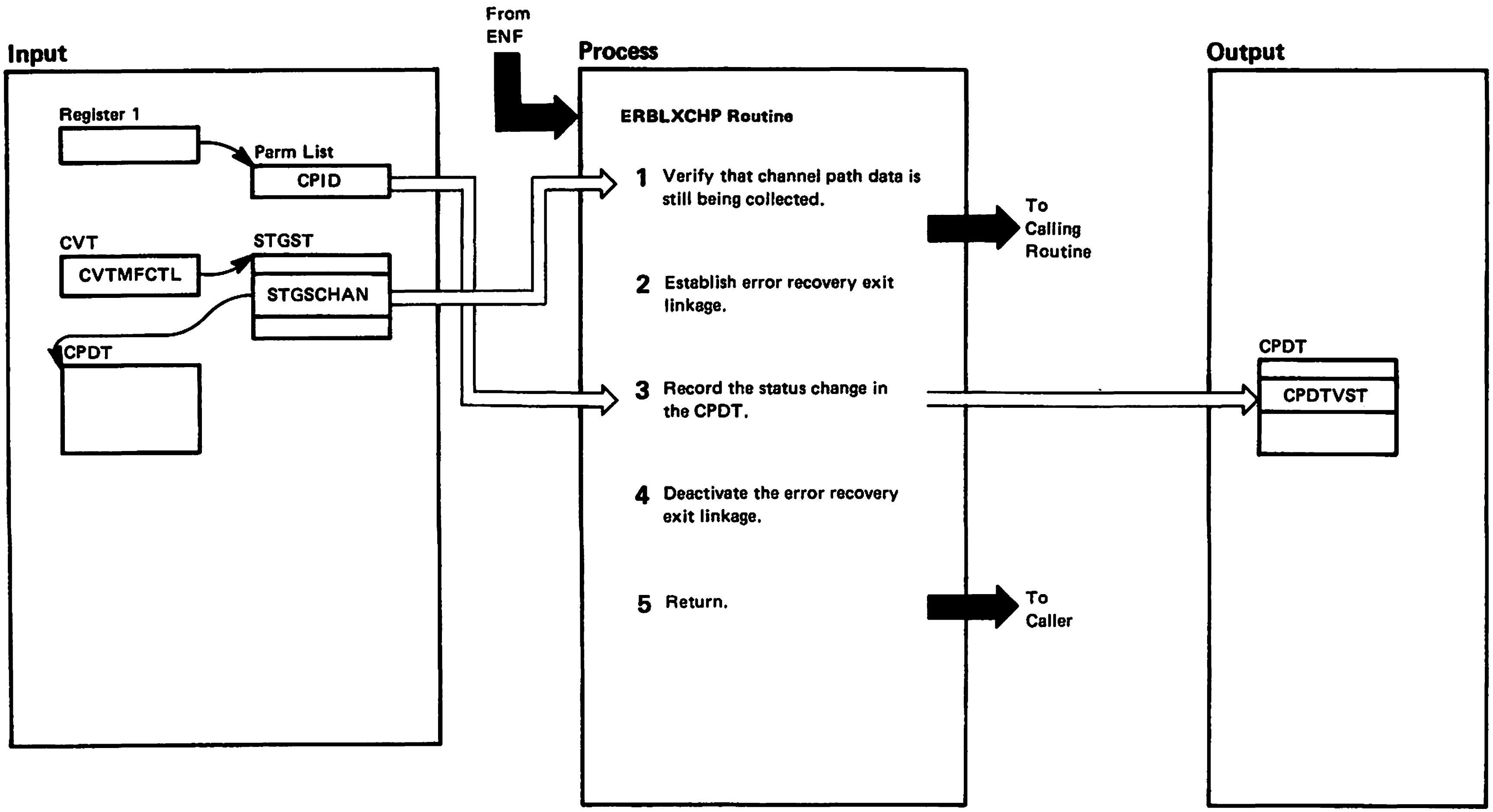
Extended Description	Module	Label
16 For a function code of 3 (retrieve next device number for a given LCU) and a device number ≦X'FFF' (the highest possible number) a loop through IODNT is started. The index into IODNT is calculated as the result of (DEVICE NUMBER+LOOP VARIABLE)//4096. If the LCU from caller matches the LCU in the IODNT, then return the associated device number in the answer field and set a return code of 00. If all IODNT entries have been searched and no match occurred or the device number was X'FFF', set a return code of 12.		
17 Return to the main routine with one of the following codes in register 15: RC=00 The information requested was retrieved and the data is available in the answer fields provided by the caller. RC=12 The search item was valid but an entry in the respective table was not found.		

Diagram 33. I/O Configuration Table Retrieve (ERBCNFGR) (Part 9 of 10)



Extended Description	Module	Label
RETRCPCU Subroutine		
18 For a function code of 4 (retrieve total number of LCUs), move the total number of LCUs from the header of the LCUT to the answer field and set a zero return code.		
19 For a function code of 6 (retrieve CPIDs and PCUs for a given LCU), use the LCU from the caller to search the LCUT. When a match is found, move the complete LCUT entry to the answer field and set a return code of 0.		
If a matching entry in the LCUT does not exist, (which occurs when the LCU is greater than or equal to the total number of LCUs in the system) set a return code of 12.		
20 Return to the main routine with one of the following codes in register 15:		
RC=00 The information requested was retrieved and the data is available in the answer fields provided by the caller.		
RC=12 The search item was valid, but an entry in the respective table was not found.		

Diagram 34. Event Listen Routine (ERBMFEAR) (Part 1 of 18)



Extended Description	Module	Label

ERBMFEAR consists of a collection of individually invoked listen exit routines called by the event notification facility (ENF). The eight listen exit routines are:

- ERBLXCHP — Channel path state change listen exit (established by channel path activity)
- ERBLXDEV — Device state change listen exit
- ERBLXDDR — DDR event listen exit
- ERBLXCMB — CMB data state change listen exit
- ERBLXCMF — CMF state change listen exit
- ERBLXCFR — Channel facility recovery event listen exit
- ERBLXIOQ — Channel path state change listen exit (established by I/O queuing activity)
- ERBLXVCP — Device path state change listen exit

ERBLXCHP Routine

ERBLXCHP listens for the event code (09) issued when a channel path is brought online or taken offline. It sets the relevant information in the channel path data table (CPDT).

1 ERBLXCHP determines if channel path data is still being collected. If not, it returns to the caller.

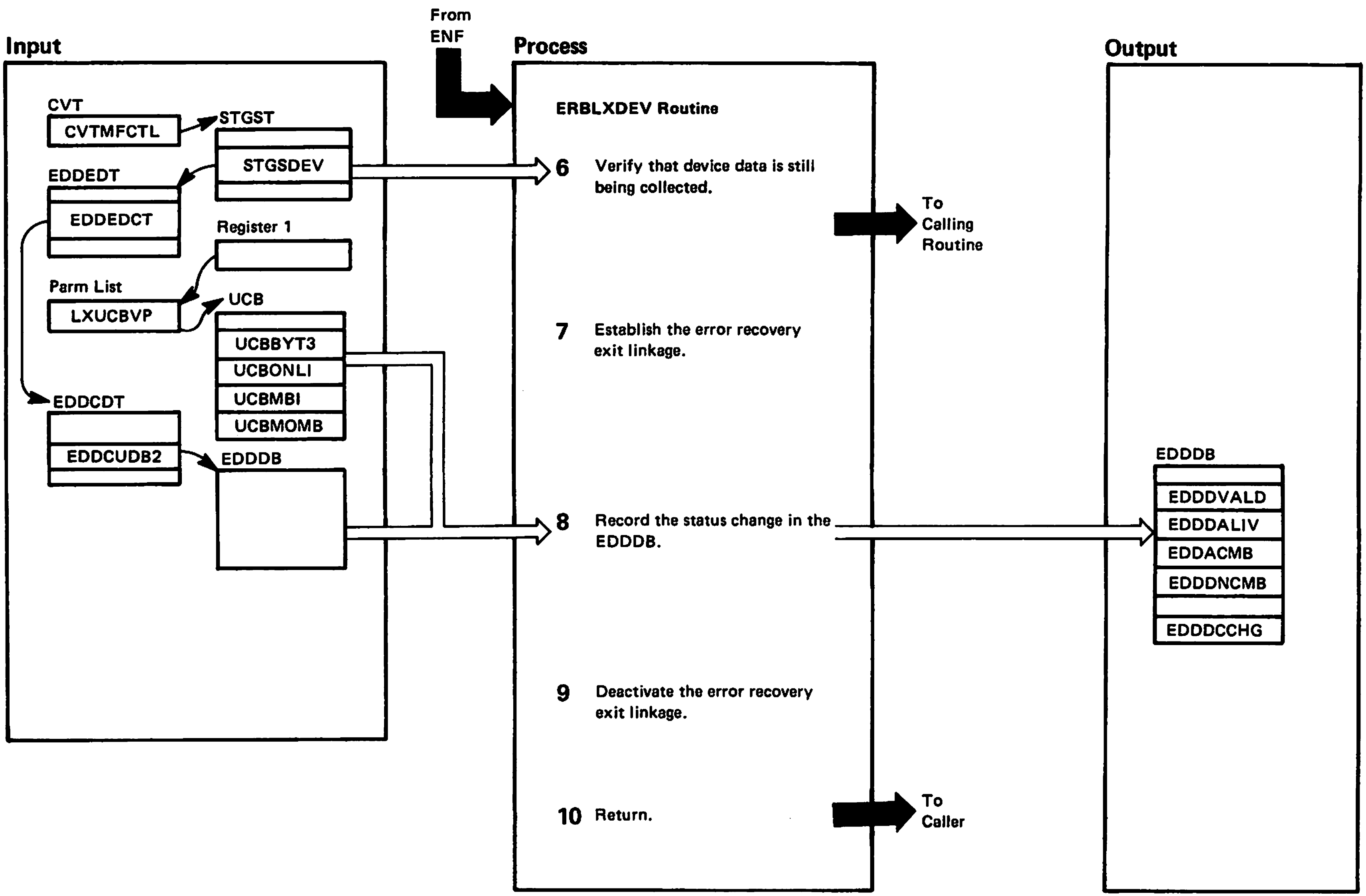
2 Issues the ESTAE macro instruction to establish a linkage to the error recovery exit (ERBLXERV).

3 ERBLXCHP uses the channel path ID as an index into the channel path table (CPDT) and sets the associated status change flag (CPDTVST).

4 Issues the ESTAE macro instruction to deactivate the error recovery exit linkage.

5 Returns to the caller.

Diagram 34. Event Listen Routine (ERBMFEAR) (Part 3 of 18)



Extended Description	Module	Label

ERBLXDEV Routine

ERBLXDEV listens for the event code (01) issued when a device is brought online or the event code (02) issued when a device is taken offline. It sets up the relevant information in the device event data block (EDDDB).

6 ERBLXDEV determines if device data is still being collected. If not, it returns to the caller. If STGSDEV contains a pointer to the device event data table (EDDEDT), device data is still being collected.

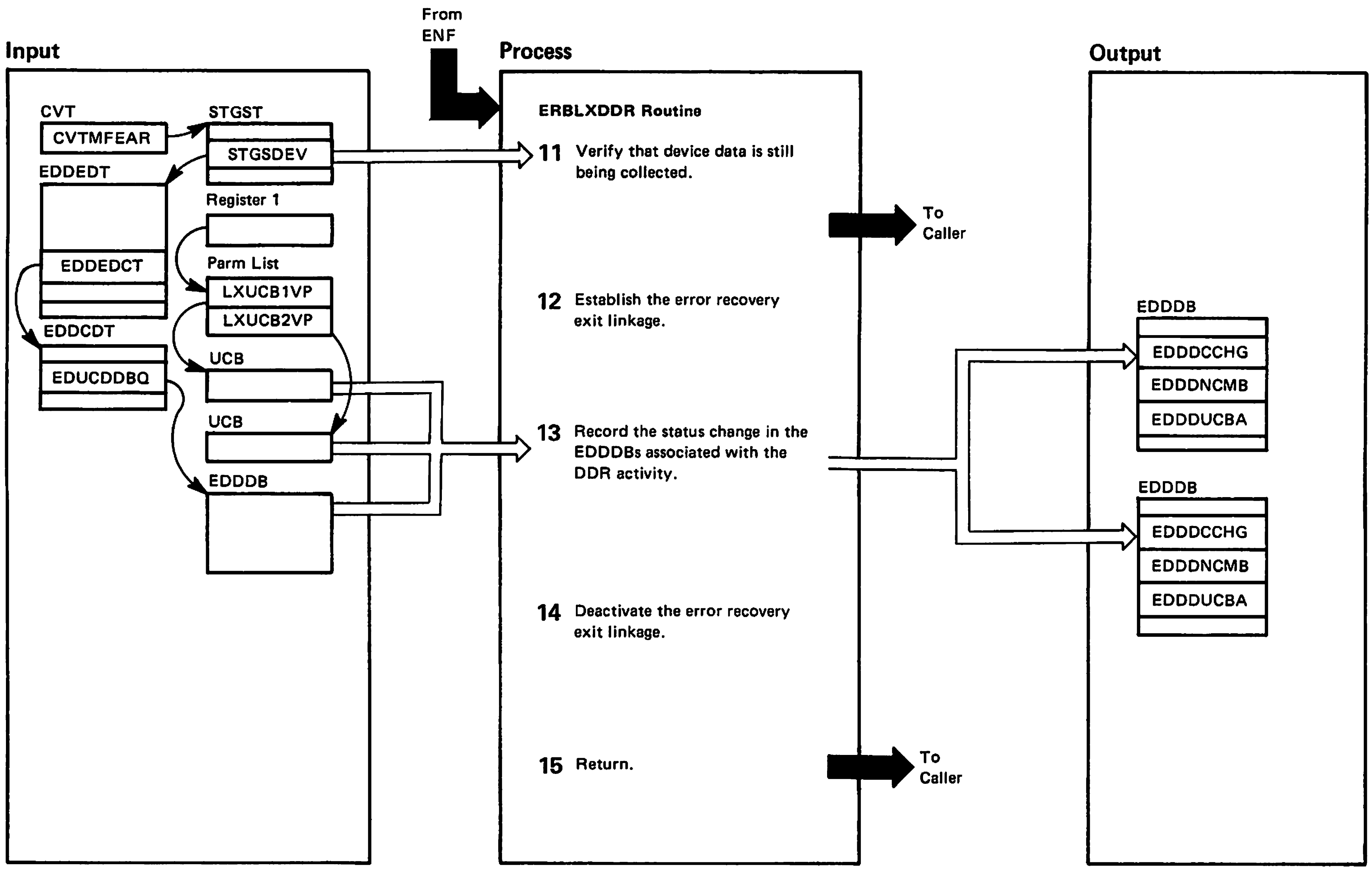
7 Issues an ESTAE macro instruction to establish linkage to the error recovery exit (ERBLXERV).

8 Finds the device data block (EDDDB) associated with the UCB whose address is contained in the input parameter. Turn on the configuration changed flag (EDDDCCHG) and check the device status (UCBONLI, UCBMBI, and UCBMM). If the device came online and measurements are active, the device alive and valid flags (EDDDALIV and EDDDVALD) and the CMB data now available flag (EDDDACMB) are set on in the device data block (EDDDB). If the device went offline, turn off the device alive flag (EDDALIV) and the CMB data now available flag (EDDDACMB) and set the no CMB data available flag (EDDDNCMB).

9 Issues the ESTAE macro instruction to deactivate the linkage to the error recovery exit.

10 Returns to the caller.

Diagram 34. Event Listen Routine (ERBMFEAR) (Part 5 of 18)



Extended Description	Module	Label

ERBLXDDR Routine

ERBLXDDR listens for the event code (10) issued when any DDR activity takes place. It sets up the relevant information in the associated device event data blocks (EDDDBs).

11 ERBLXDDR determines if device data is still being collected. If not, it returns to the caller. If STGSDEV contains a pointer to the device event data table (EDDEDT) device data is still being collected.

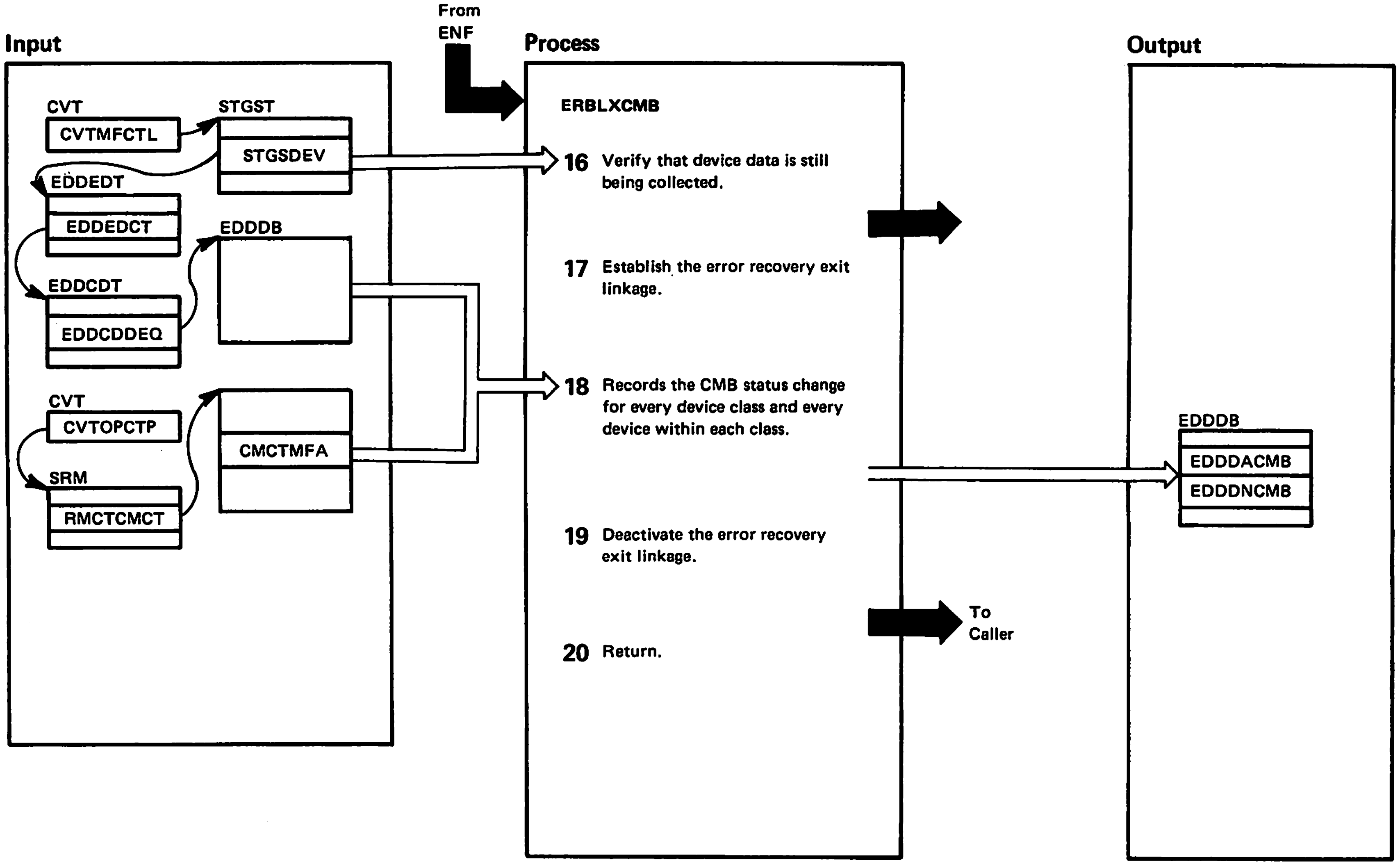
12 Issues an ESTAE macro instruction to establish the linkage to the error recovery exit (ERBLXERV).

13 ERBLXDDR finds the device data blocks associated with the two UCBs involved, sets the configuration changed flag (EDDDCCHG) and no CMB data available flag (EDDDNCMB), and places the new device numbers in the device data blocks. If both device blocks are found, ERBLXDDR sets 'configuration changed' and 'no CMB data available' flags in both EDDDBs then switches the UCB addresses. If only one device block is found, it sets the 'configuration changed' and 'no CMB data available' flags in the EDDDB and inserts the other UCB address.

14 Issues the ESTAE macro instruction to deactivate the error recovery exit linkage.

15 Returns to the caller.

Diagram 34. Event Listen Routine (ERBMFEAR) (Part 7 of 18)



Extended Description	Module	Label

ERBLXCMF Routine

ERBLXCMF listens for event code (06) which signals CMF data state changes. It sets up the relevant information in the I/O queuing event data table.

21 ERBLXCMF determines if I/O queuing data is still being collected. If not, it returns to the caller. If STGSIOQ contains a pointer to the I/O queuing event data table (EIOQED), data is being collected.

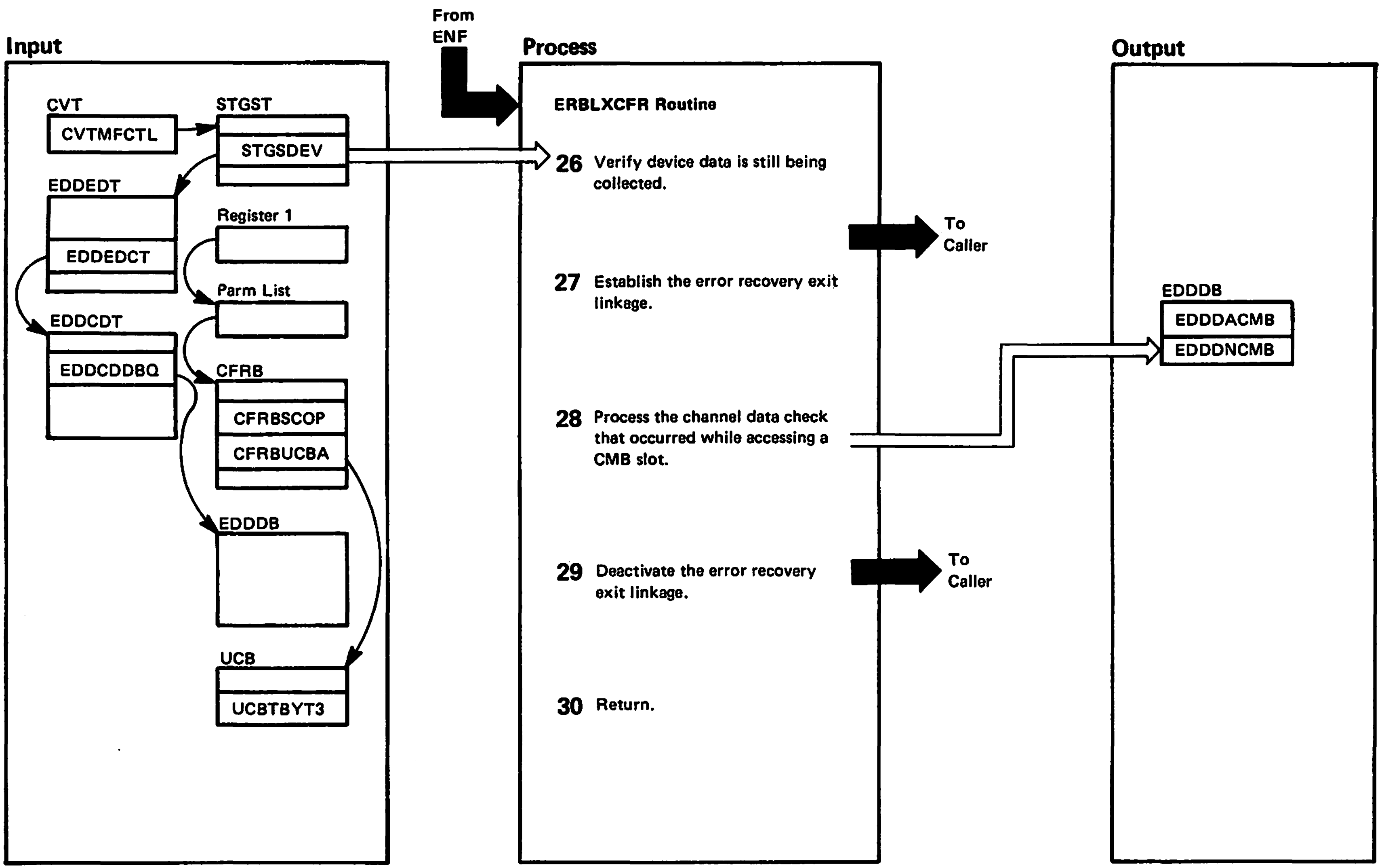
22 Issues an ESTAE macro instruction to establish the linkage to the error recovery exit (ERBLXERV).

23 If the channel measurement facility (CMF) is not operating (CMCTMA=0), ERBLXCMF turns off the CMF data available flag (EIOQACMF) and sets on the no CMF data available flag (EIOQNCMF). Otherwise, it sets the CMF data available flag on.

24 Issues the ESTAE macro instruction to deactivate the error recovery exit linkage.

25 Returns to the caller.

Diagram 34. Event Listen Routine (ERBMFEAR) (Part 11 of 18)



Extended Description	Module	Label

ERBLXCFR Routine

ERBLXCFR listens for the event code (11) issued when a channel data check occurred while accessing a CMB slot. If the UCB involved indicates a non-TAPE/DASD device, an SDUMP is provided. (For TAPE/DASD devices, SRM has already provided an SDUMP).

26 ERBLXCFR determines if device data is still being collected. If not, it returns to the caller. If STGSDEV contains a pointer to the device event data table (EDDEDT), data is being collected.

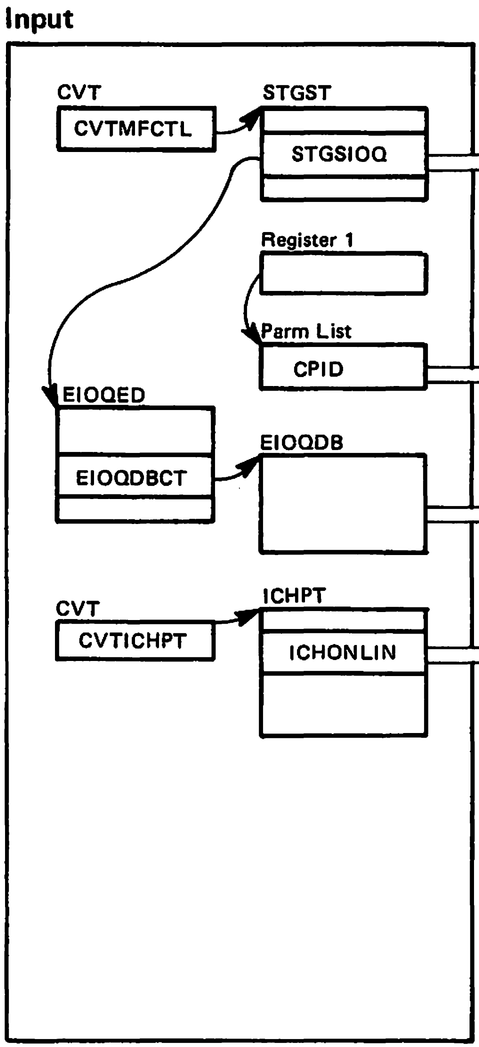
27 Issues an ESTAE macro instruction to establish the linkage to the error recovery exit (ERBLXERV).

28 If a UCB address is provided with the channel facility recovery block (CFRB) indicating that the scope of the error is local, ERBLXCFR determines the associated device class. If the device class is non-TAPE/DASD and if the device data block is found, ERBLXCFR turns off the CMB data available flag (EDDDACMB) and sets on the CMB data not available flag (EDDDNCMB), and issues the SDUMP macro instruction. The SDUMP contains the same data the SRM provides for TAPE/DASD devices.

29 Issues the ESTAE macro instruction to deactivate the error recovery exit linkage.

30 Returns to the caller.

Diagram 34. Event Listen Routine (ERBMFEAR) (Part 13 of 18)



From ENF

Process

ERBLXIOQ Routine

31 Verify that I/O queuing data is still being collected.

To Caller

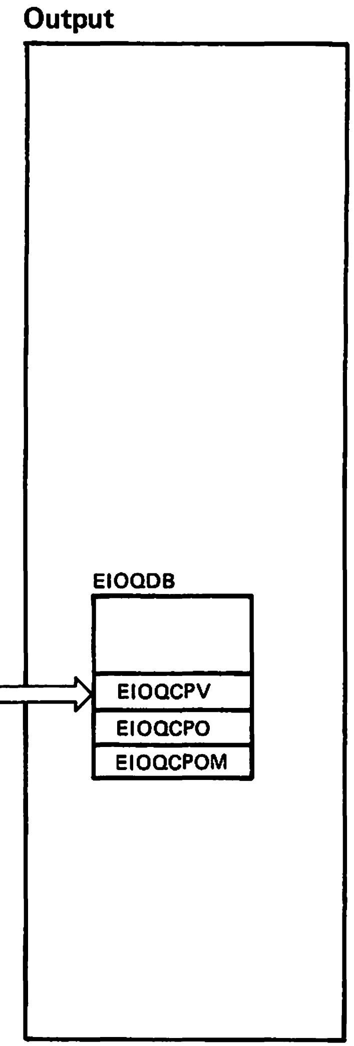
32 Establish the error recovery exit linkage.

33 Record the channel path status change in the EIOQDB.

34 Deactivate the error recovery exit linkage.

35 Return.

To Caller



Extended Description	Module	Label

ERBLXIOQ Routine

ERBLXIOQ listens for the event code (09) issued when a channel path moves online or offline. It updates the relevant configuration information in the I/O queuing data blocks (EIOQDBs) involved.

31 ERBLXIOQ determines if I/O queuing data is still being collected. If not, it returns to the caller. If STGSIOQ contains a pointer to the I/O queuing event data table (IOQED), data is being collected.

32 Issues an ESTAE macro instruction to establish the linkage to the error recovery exit (ERBLXERV).

33 For each LCU to which the CPID belongs, update the following channel path information.

- Set the vary activity flag (EIOQCPV).
- Set the online/offline flag according to the current status of the channel path in the ICMPT (EIOQCPO).
- Update the channel path online mask is updated accordingly (EIOQCPOM).

34 Issues the ESTAE macro instruction to deactivate the error recovery exit linkage.

35 Returns to caller.

Diagram 34. Event Listen Routine (ERBMFEAR) (Part 15 of 18)

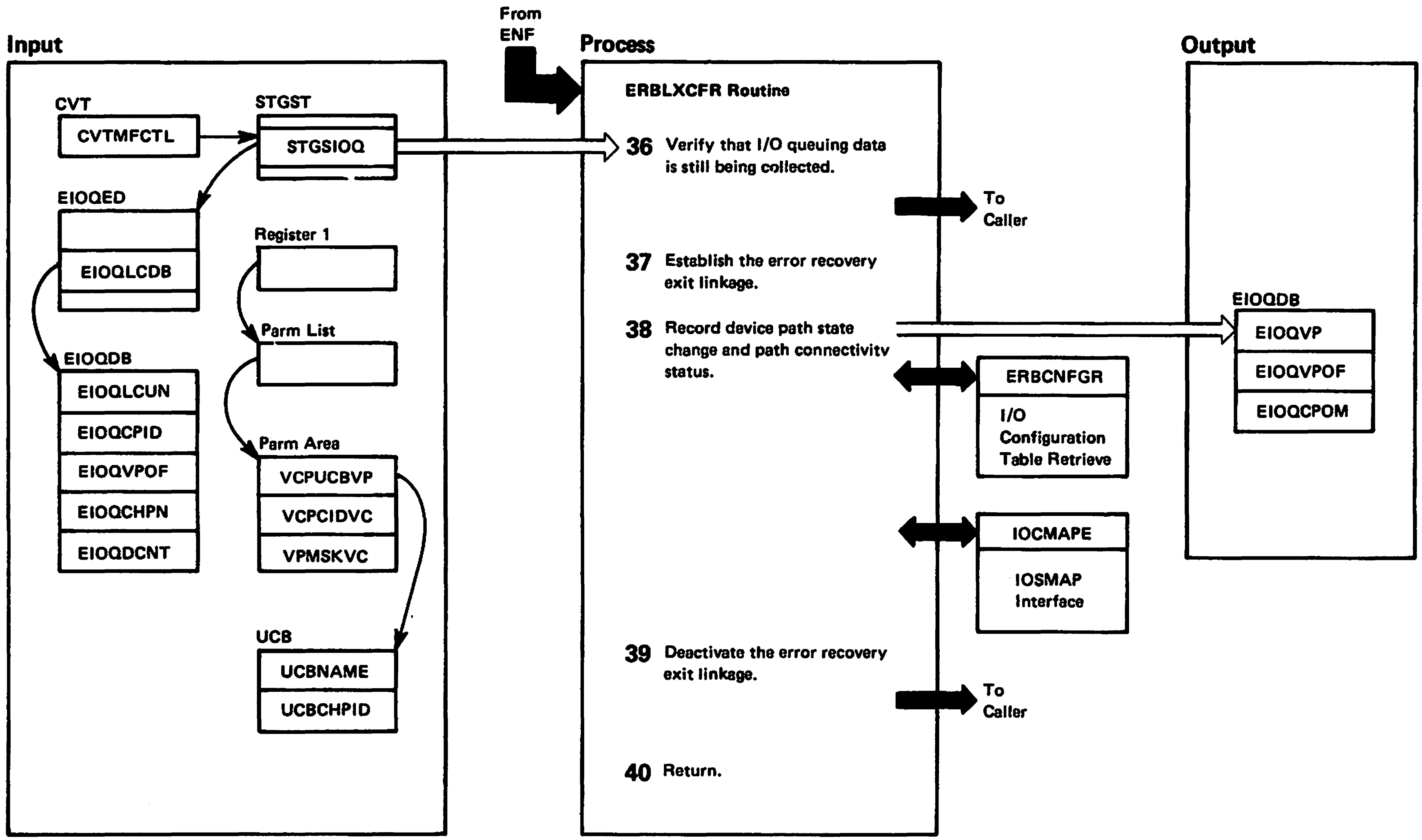


Diagram 34. Event Listen Routine (ERBMFEAR) (Part 16 of 18)

Extended Description	Module	Label

ERBLXVCP Routine

ERBLXVCP listens for the event code (08) issued when a channel path moves online or offline. It updates the relevant configuration information in the I/O queuing data blocks (EIOQDBs) involved.

36 ERBLXVCP determines if I/O queuing data is still being collected. If not, it returns to the caller. If STGSIOQ contains a pointer to the I/O queuing event data table (IOQED), data is being collected.

37 Issues an ESTAE macro instruction to establish the linkage to the error recovery exit (ERBLXERV).

38 ERBLXVCP records device path state change and path connectivity status. ERBLXVCP invokes ERBCNFGR to determine the LCU for the device path. For each device in this LCU, it invokes the IOSMAP interface to get the logical path mask (LPM) and issues an IOSLOOK to obtain the associated UCB. If the connectivity status has changed, it sets on the vary path indicator (EIOQVP) and depending on whether the path came online or went offline, sets the channel path online mask (EIOQCPOM) and the path offline mask (EIOQVPOF).

39 Issues the ESTAE macro instruction to deactivate the error recovery exit linkage.

40 Returns to caller.

Diagram 34. Event Listen Routine (ERBMFEAR) (Part 17 of 18)

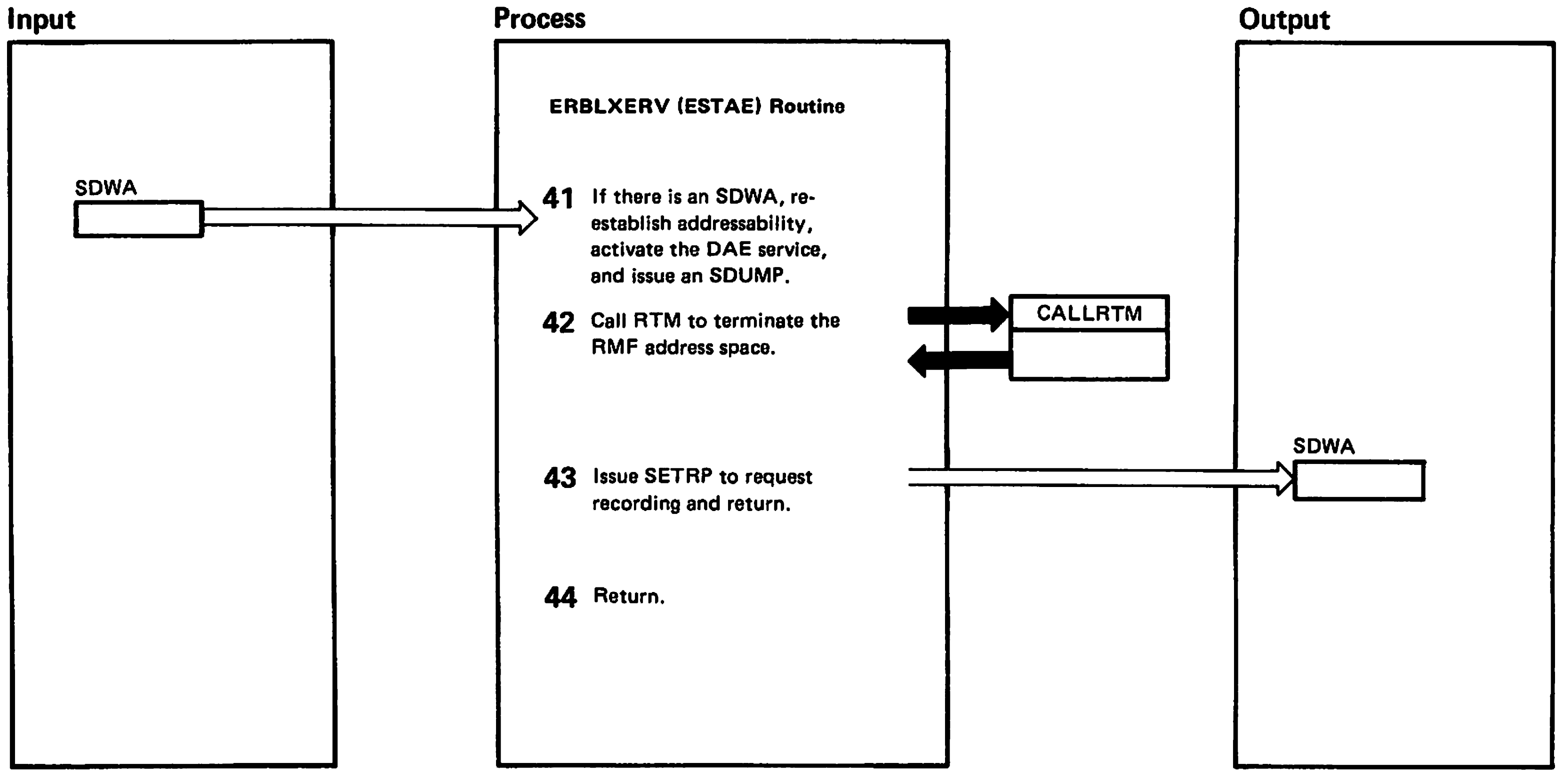


Diagram 28. Event Listen Routine (ERBMFEAR) (Part 18 of 18)

Extended Description	Module	Label
41 If an SDWA exists, gets the address of the base register and data from the ESTAE parameter list. If SDWAEAS=0 sets up information for the SDWA, activates the DAE service, and issues the SDUMP macro instruction.		
42 Issues the CALLRTM macro to request termination of the RMF address space. Issues the 'OFE' completion code.		
43 Issues the SETRP macro instruction to request error recording and return.		
44 Returns to the caller.		

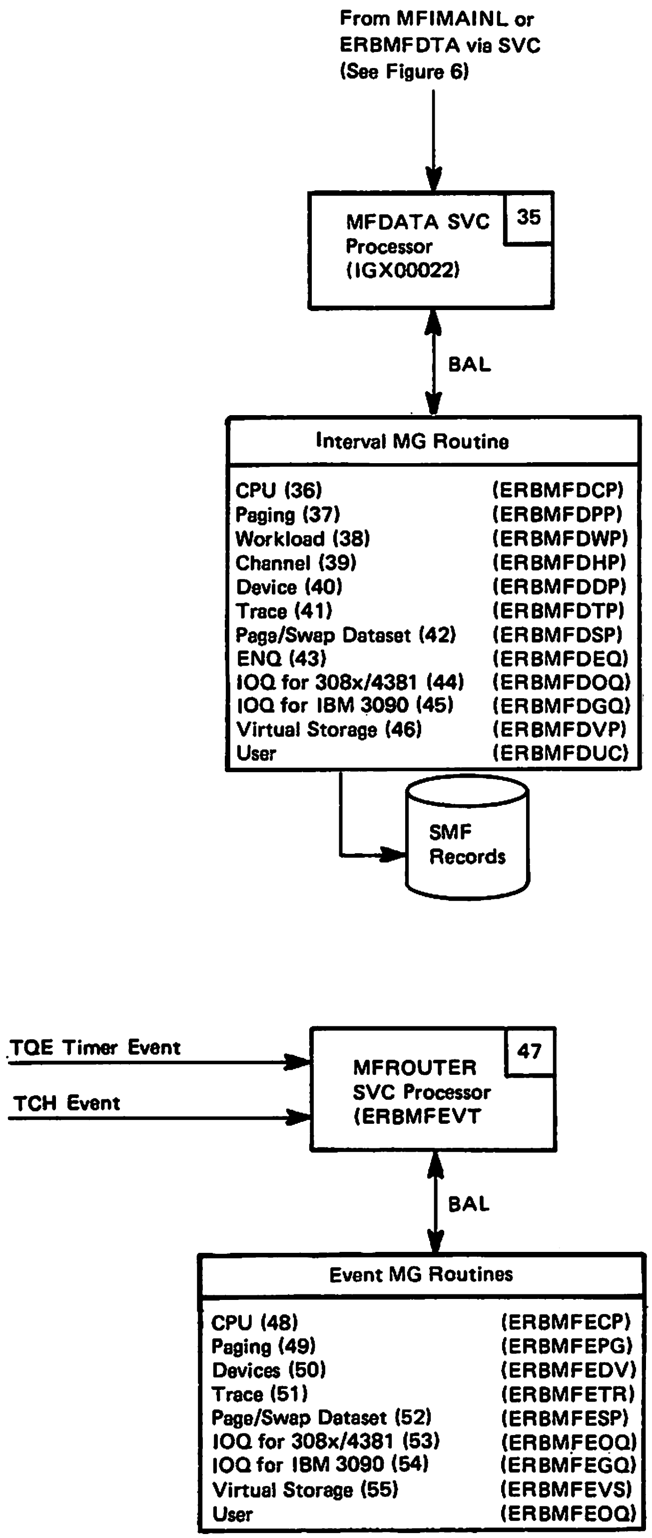


Figure 7. Monitor I Data Gathering Overview

Diagram 35. MFDATA SVC Mainline Processor (IGX00022) (Part 1 of 4)

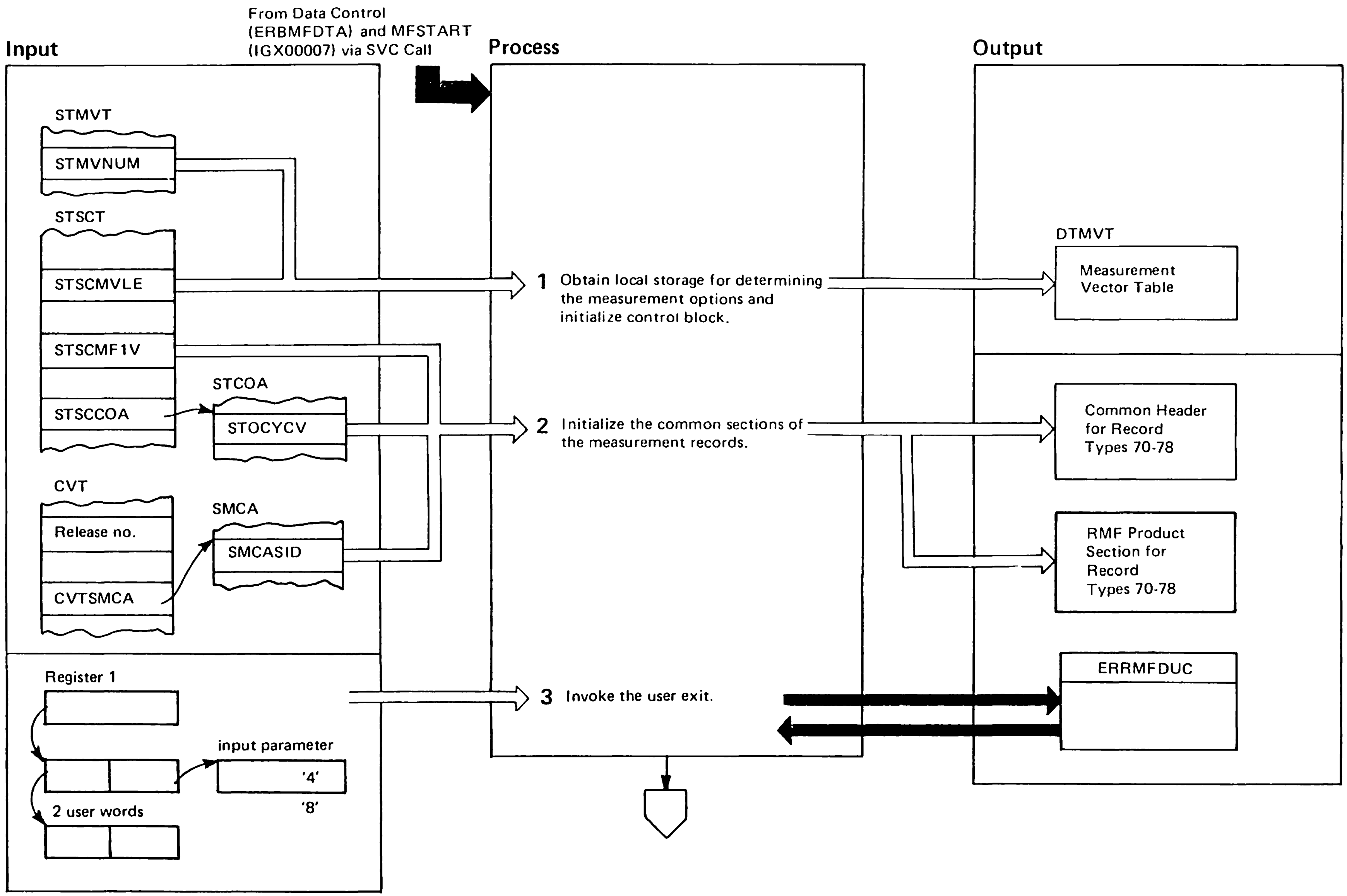


Diagram 35. MFDATA SVC Mainline Processor (IGX00022) (Part 2 of 4)

Extended Description	Module	Label
The MFDATA SVC Mainline (IGX00022) processor executes in response to an MFDATA SVC issued by the Data Control module (ERBMFDTA), once each interval, and by MFSTART (IGX00007) during initialization. When called, IGX00022 controls the operation of measurement gathering routines. Each MG routine collects measurements of one of the following kinds if called for by input option:	IGX00022	

- CPU wait time
- Paging activity
- Workload
- Channel activity
- Device activity
- ASM/RSM/SRM Trace activity
- Page/Swap dataset activity
- ENQ activity
- I/O Queuing activity

The measurements for the interval are placed in records that have the format of System Management Facilities (SMF 70-78). Internal Copies of these records are used by report generation routines (SARG) to provide printed reports specified by input options.

Extended Description	Module	Label
1 Issue the GETMAIN macro instruction to obtain storage for the Measurement Vector Table (DTMVT) and initialize the table area by setting all option pointers to zero.	IGX00022	

2 Obtain SMF record header items for:

- Identifying the record as an MVS/XA record.
- System identification.
- Subsystem indicator (RMF).
- Subtype.
- Length and count of product section.

Obtain RMF Product section items for:

- RMF Version number
- Cycle length (from input options).
- Product name (RMF).
- MVS software level.
- RMF I/O measurement level.

Extended Description	Module	Label
3 If the EXITS option is on, stores the module/CSECT name of the user data gathering exit in ERBM1RCB and invokes the user exit. On return, restores the contents of ERBM1RCB.		

- Input parameters '4'x indicates ZZ initialization.
- '8'x indicates the end of an RMF interval.

Diagram 35. MFDATA SVC Mainline Processor (IGX00022) (Part 3 of 4)

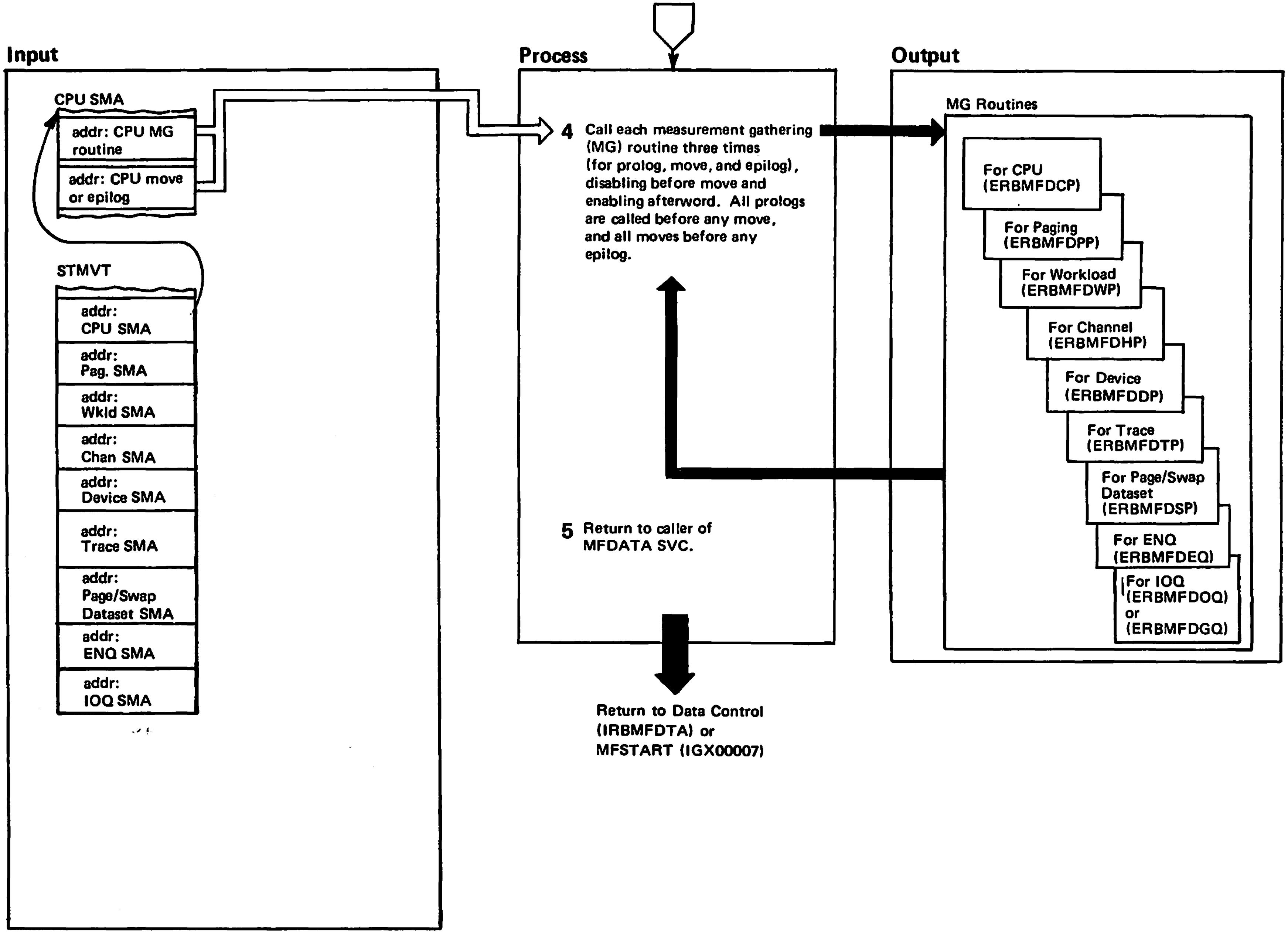


Diagram 35. MFDATA SVC Mainline Processor (IGX00022) (Part 4 of 4)

Extended Description	Module	Label
4 For recovery purposes, MFDATA SVC stores in ERBM1RCB the module/CSECT name of any MG routine that it calls.		
Each MG routine has a prolog, a move part, and an epilog. The prologs for all the required (by input option) MG routines are called first in the order listed in the first paragraph of this explanation. When the prologs have been called, the required move parts are called, and then the epilogs are called. The effect on each MG routine, however, is as though it executed from start to end without interruption. This arrangement is used to allow the move parts of these routines and IGX00022 to execute disabled. Before the move parts of the MG routines, which contain the code to move measurement data into record formats, are executed, interruptions are disabled by obtaining and releasing the dispatcher lock. When the SETLOCK is released, it is released disabled. The reverse technique is used to enable, after all the move parts of the MG routines have been executed.	ERBMFDCP ERBMFDPP ERBMFDWP ERBMFDHP ERBMFDDP ERBMFDTP ERBMFDSP ERBMFDEQ ERBMFDOQ ERBMFDGQ	
5 Upon return to the caller, IGX00022 saves the measurement vector table (DTMVT) address in register 1 and restores its own module/CSECT name in ERBM1RCB.	IGX00022	

Diagram 36. Interval MG Routine for CPU (ERBMFDCP) (Part 1 of 6)

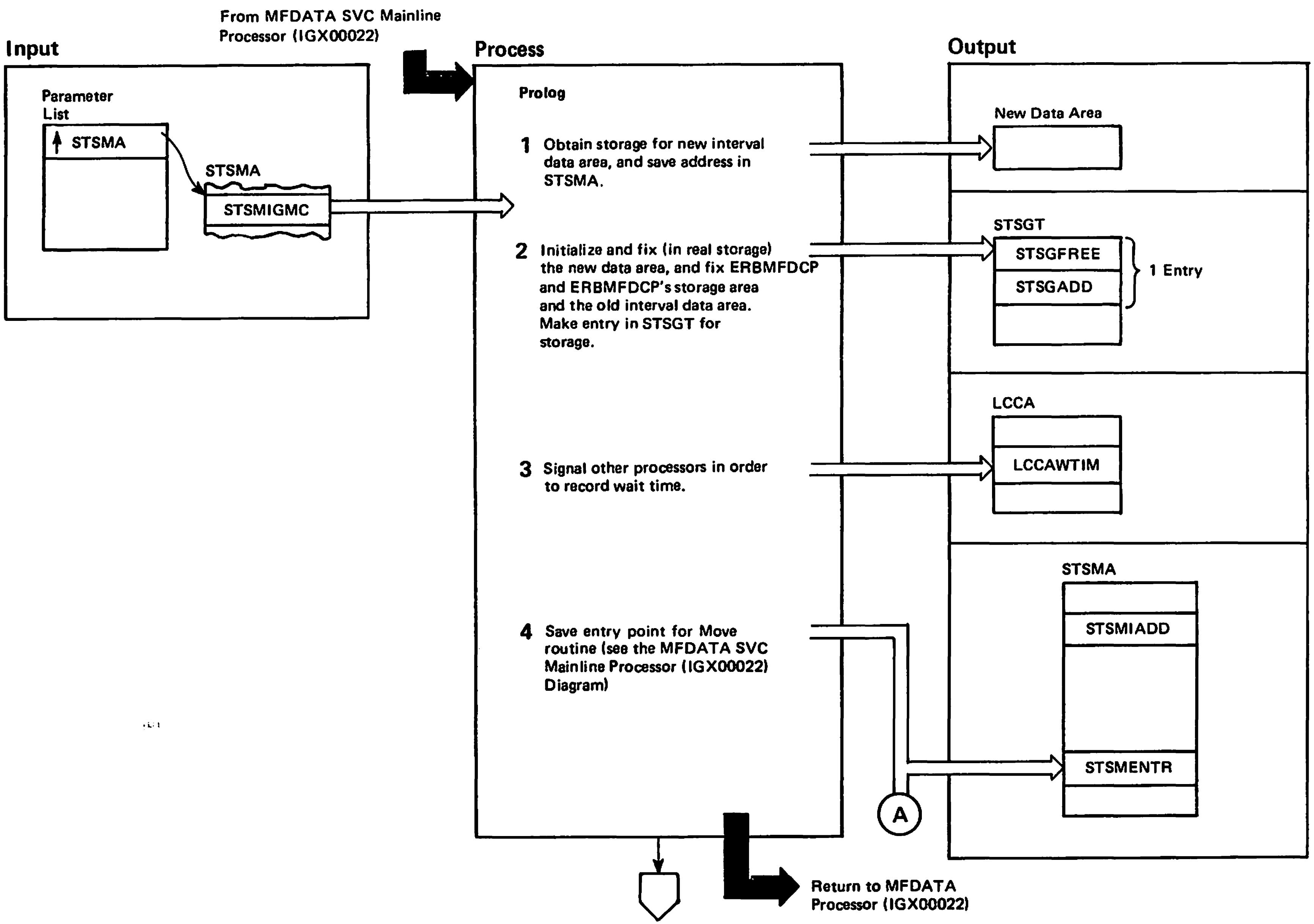


Diagram 36. Interval MG Routine for CPU (ERBMFDCP) (Part 2 of 6)

Extended Description	Module	Label
The Interval MG Routine for CPU (ERBMFDCP) receives control from the MFDATA SVC Processor at the end of each interval if CPU activity reports are required. ERBMFDCP copies CPU wait times, the number of entries to the I/O SLIH, and the number of TPI instructions with CC=1 for all CPUs into a contiguous storage area and builds an internal image of the RMF CPU activity record (SMF70HDR) for the SMF data set. ERBMFDCP calculates wait time, the number of entries to the I/O SLIH, and the number of TPI instructions with CC=1 for each CPU by subtracting the corresponding value read at the end of the current interval from that read at the end of the previous interval, after adjusting for the possibility of wrap-around readings.	ERBMFDCP	
Prolog		
1 Use the GETMAIN macro instruction to obtain the required storage in key zero. Save the address in the STSMIADD field of the Supervisor Measurement Area (STSMA).	ERBMFDCP	DCGETMN1
2 Store the subpool and length of the storage obtained into the first word of the area. Use the PGSER macro instruction to fix the old and new interval data areas ERBMFDCP, and ERBMFDCP's storage area. Save the storage area's subpool, length, and address in an entry in the Storage Resource Table (STSGT).		
3 Use the RISGNL macro instruction to signal other CPU's out of the wait state so that accumulated wait time will be recorded in the LCCA. This will be used later by ERBMFDCP to compute the interval wait time.		
4 Save the entry point, as described in the M.O. diagram MFDATA SVC Mainline Processor (IGX00022), for use in returning to the Move part of ERBMFDCP.	ERBMFDCP	

Diagram 36. Interval MG Routine for CPU (ERBMFDCP) (Part 3 of 6)

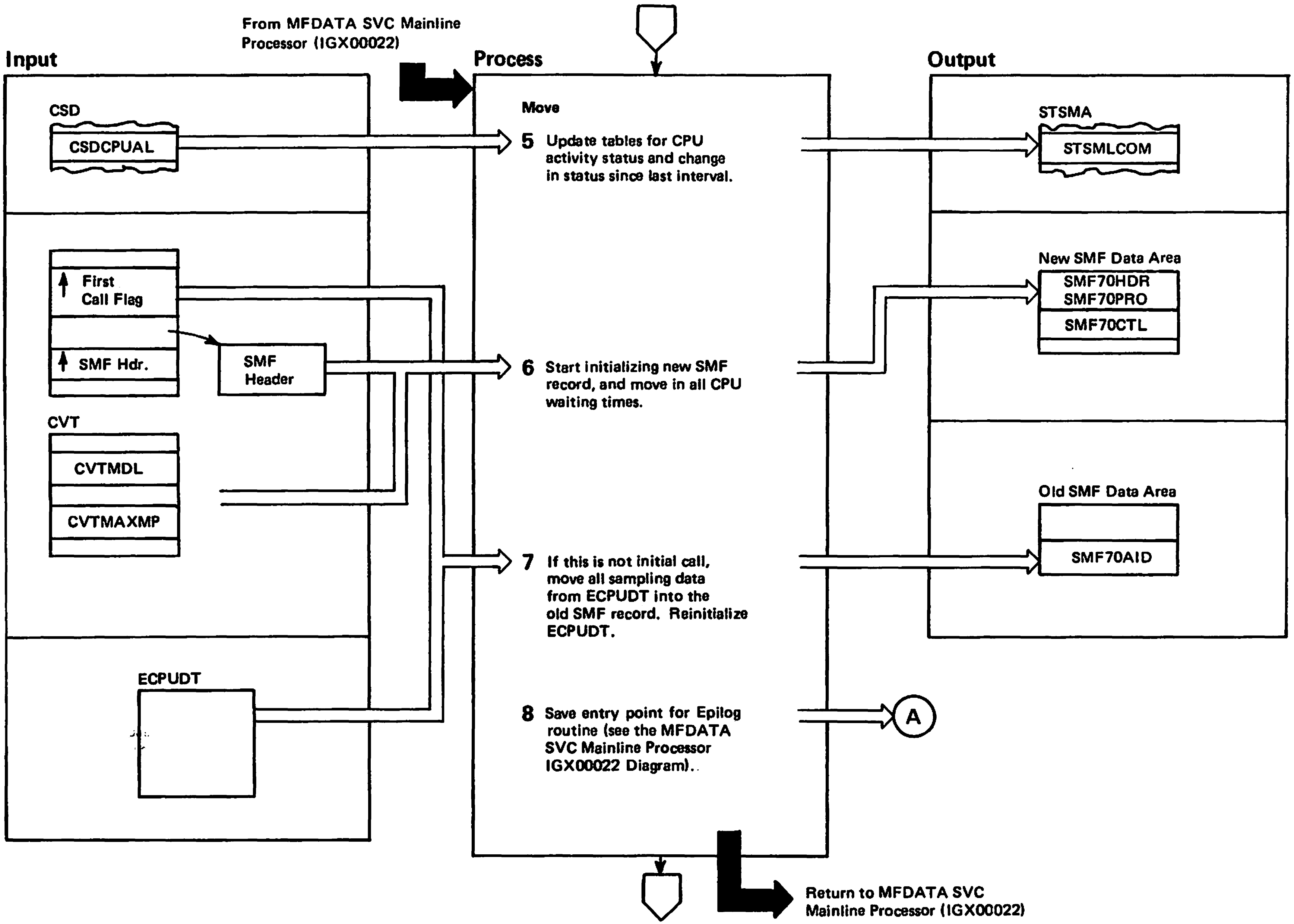


Diagram 36. Interval MG Routine for CPU (ERBMFDCP) (Part 4 of 6)

Extended Description	Module	Label
Move		
5 If a CPU is now online whose flag is not set in STSMLCCM of the Supervisor Measurement Area (STSMA), set its flag to indicate that it has been online.	ERBMFDCP	DCMOVE
6 Partially initialize the new SMF record by moving in standard SMF header, product section and setting the following fields: a. All offset/length/number triplets are calculated and initialized for: RMF product section CPU control section CPU data sections ASID data section b. record type c. CPU model number d. zero control portion Set online status flags for all valid CPU's, and move wrap-around wait time measurement counters for those valid CPU's into the new SMF record.		
7 If this is initial call, no samples will have been taken yet, and only one SMF record will exist. If not initial call, move all sampling data from the CPU event data table (ECPUDT) into the old SMF record. Zero out ECPUDT and set minimum values to a high number.		
8 See step 3.	ERBMFDCP	DDEPILOG

Diagram 36. Interval MG Routine for CPU (ERBMFDCP) (Part 5 of 6)

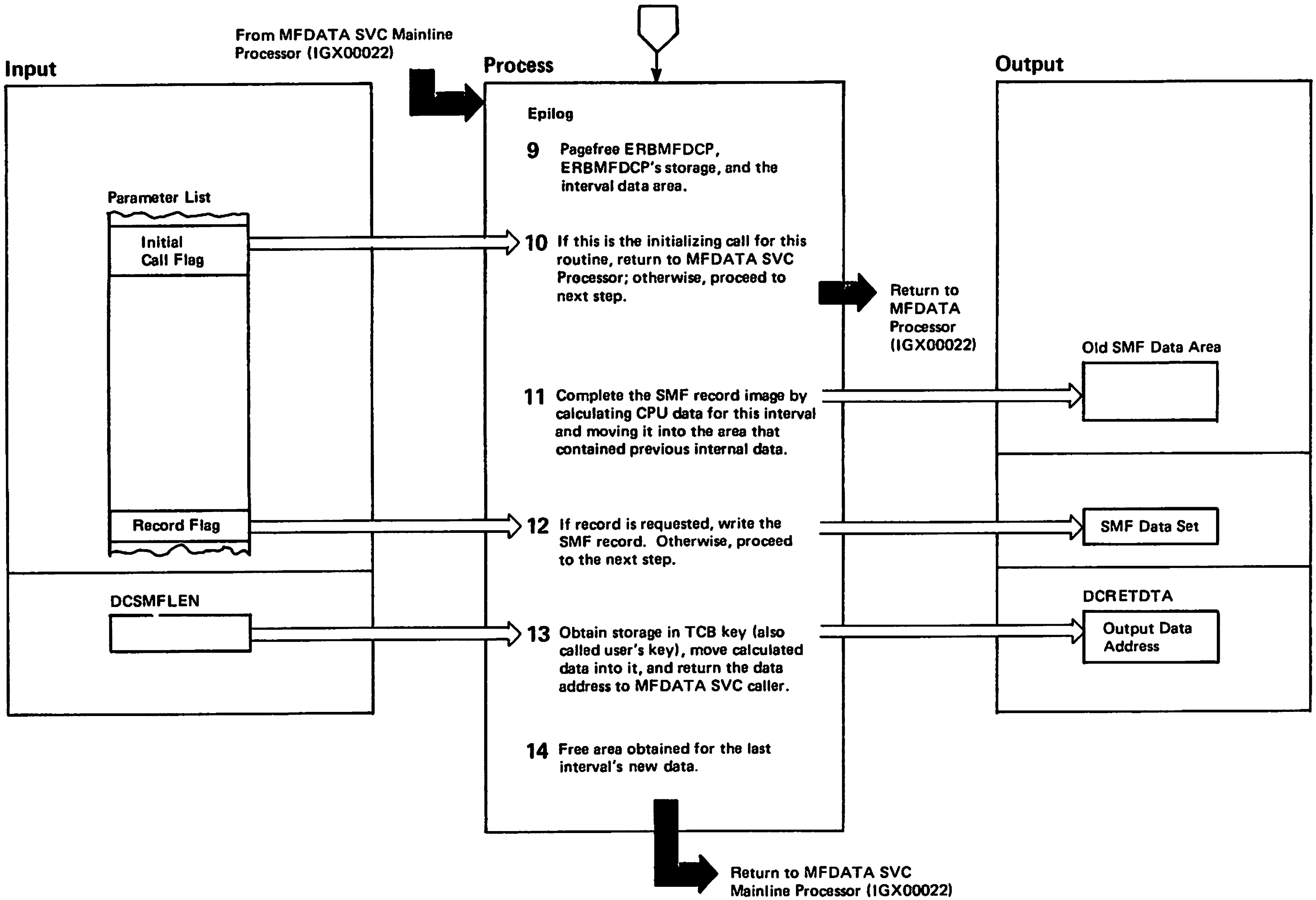
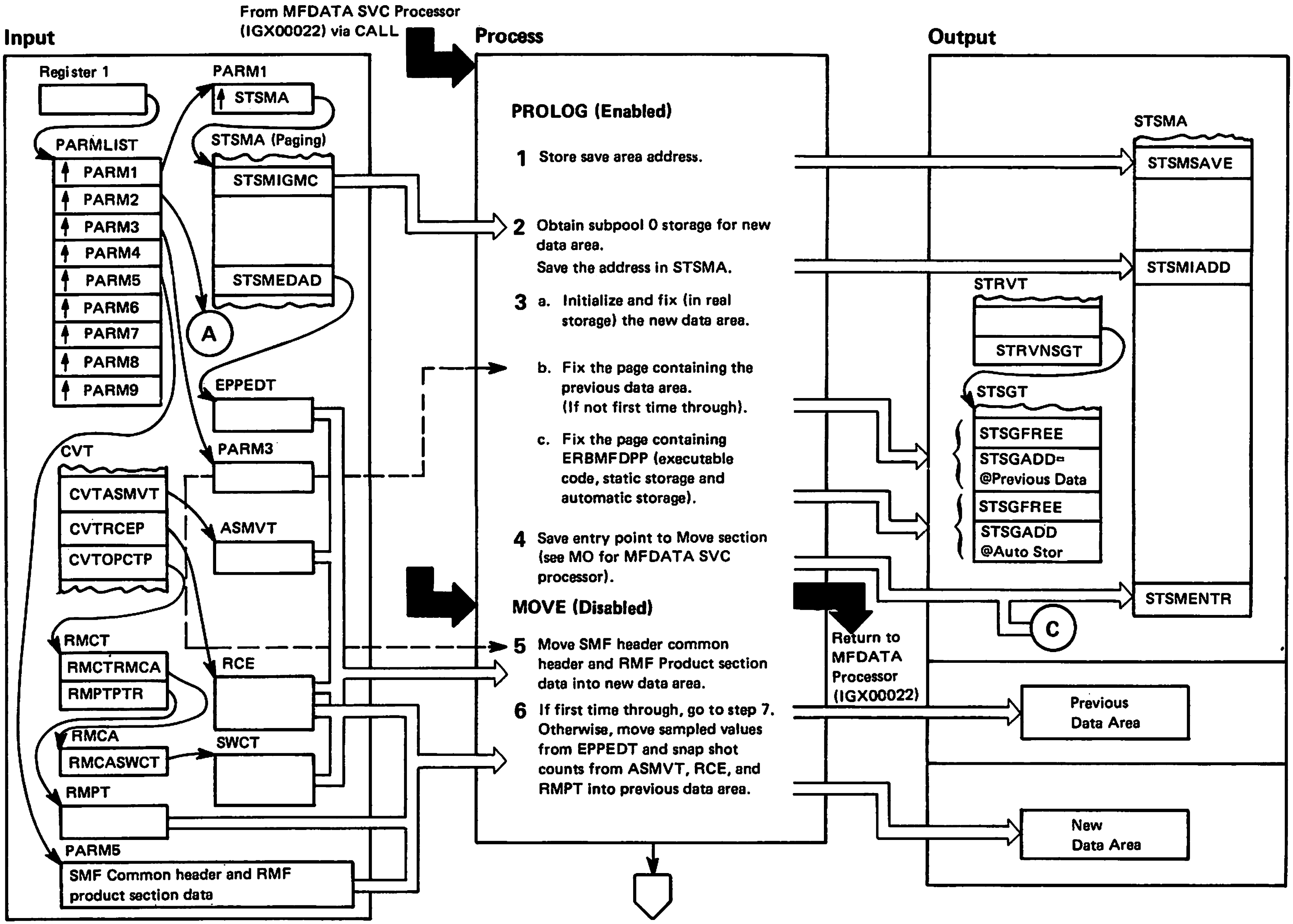


Diagram 36. Interval MG Routine for CPU (ERBMFDCP) (Part 6 of 6)

Extended Description	Module	Label
9 Use the PGSER macro instruction to free ERBMFDCP, ERBMFDCP's storage, and the interval data area.		
10 On the first call to the MFDATA SVC, the MFDATA SVC Processor calls the interval MG routines to obtain a first set of wrap-around measurements for later calculations (subtraction).	ERBMFDCP	
11 Move through all possible CPU entries in old and new data areas, and calculate CPU wait times, the number of entries to the I/O SLIH, and the number of TPI instruction with CC=1 for CPUs active throughout the interval. Allow for wrap-around values when subtracting current from previous values.	ERBMFDCP	
12 Use the SMFWTM macro instruction to write the image of the SMF70HDR record to the SMF data set.	ERBMFDCP	
13 Use the GETMAIN macro instruction to obtain the required storage in user key; use the MODESET macro instruction to change to the TCB key.	ERBMFDCP	
14 Release the storage of the interal SMF image using a FREEMAIN macro instruction.	ERBMFDCP	

Diagram 37. Interval MG Routine for Paging (ERBMFDPP) (Part 1 of 6)



Extended Description	Module	Label
The Interval MG Routine for Paging (ERBMFDPP) builds an internal image of an SMF-71 paging record. If the RECORD option was specified when RMF was started, it writes this image to the SMF data set. For the internal image, ERBMFDPP uses data collected by the real storage manager, the paging supervisor and the auxiliary storage manager. As described in the M.O. for the MFDATA SVC Processor, ERBMFDPP executes in three parts, PROLOG, MOVE, and EPILOG, but no break in execution is apparent except for the need to save entry points for the MOVE and EPILOG parts.		
PROLOG		
1 On entry, the address of a standard save area is stored into the STSMSAVE field of the supervisor measurement area (STSMA). The address is later passed as a parameter on subsequent calls by IGX00022 when it reenters at the MOVE and EPILOG sections.	ERBMFDPP	
2 The GETMAIN macro instruction is used to obtain subpool 0 storage for a new data area. Data for the next interval (the one just starting) is to be collected in this storage. The address of the acquired storage is saved in the STSMIADD field of the STSMA after the previous data area's address is first saved. Data for the current interval (the one just ending) is collected in this storage.	ERBMFDPP	
3 ERBMFDPP uses the PGSER macro with the fix parameter to inhibit paging of the new data area, the previous data area (if this is not the first time through this step and therefore the previous data area already exists), the ERBMFDPP routine itself, its local storage, and the ERBMFDPP routine's automatic storage. Entries are made in the storage resource table (STSGT) for the previous data area and ERBMFDPP's automatic storage. (The subpool and length are in the STSGFREE field, and the address is in the STSGADD field). The index in the STRVNSGT field of the resource vector table (STRVT) is incremented so that the next STSGT entry is available for use.	ERBMFDPP	
Note, page fixing is done to prohibit page fault interrupts during the disabled MOVE section.		

Extended Description	Module	Label
4 ERBMFDPP saves the entry point in the STSMA to be used by IGX00022 to enter the MOVE section at step 5. Between the PROLOG and MOVE sections a return is made to the caller that avoids freeing data that would be freed in a normal return.	ERBMFDPP	
MOVE		
5 ERBMFDPP initializes the internal record image for the next interval (the one just starting) in the new data area with the following data:	ERBMFDPP	
a. SMF common header and RMF product sections.		
b. Offset/length/count triplets in the entered header.		
c. Paging data fields — start-of-interval values for wrap around fields that reflect cumulative counts for pages in, pages out, pages reclaimed, swap in, swap out, no pages in, no pages out, no pages reused, no pages moved to extended storage, and swap counts per swap reason.		
6 If this is the first time through (as indicated by PARM3), ERBMFDPP skips this step. Otherwise, ERBMFDPP fills in the following paging data fields of the internal record image (SMF71HDR) being built for this interval (the one just ending) in the previous data area:	ERBMFDPP	
a. End-of-interval snapshot values for the user pool slot usage are obtained from the auxiliary storage management vector table (ASMVT).		
b. End-of-interval snapshot values for frame usage are obtained from the RSM control and enumeration area (RCE).		
c. End-of-interval a snapshot of the SRM OPT member that is currently in use is obtained from the RSM parameter table (RMPT).		
d. End-of-interval sampled values of minimum, maximum, and average frame and slot usage counts are obtained from the paging event data table (EPPEDT). The number of samples is stored in the header of the SMF record.		
e. The nucleus frame count and the number of installed extended storage frames are obtained from values gathered at initialization.		

Diagram 37. Interval MG Routine for Paging (ERBMFDPP) (Part 3 of 6)

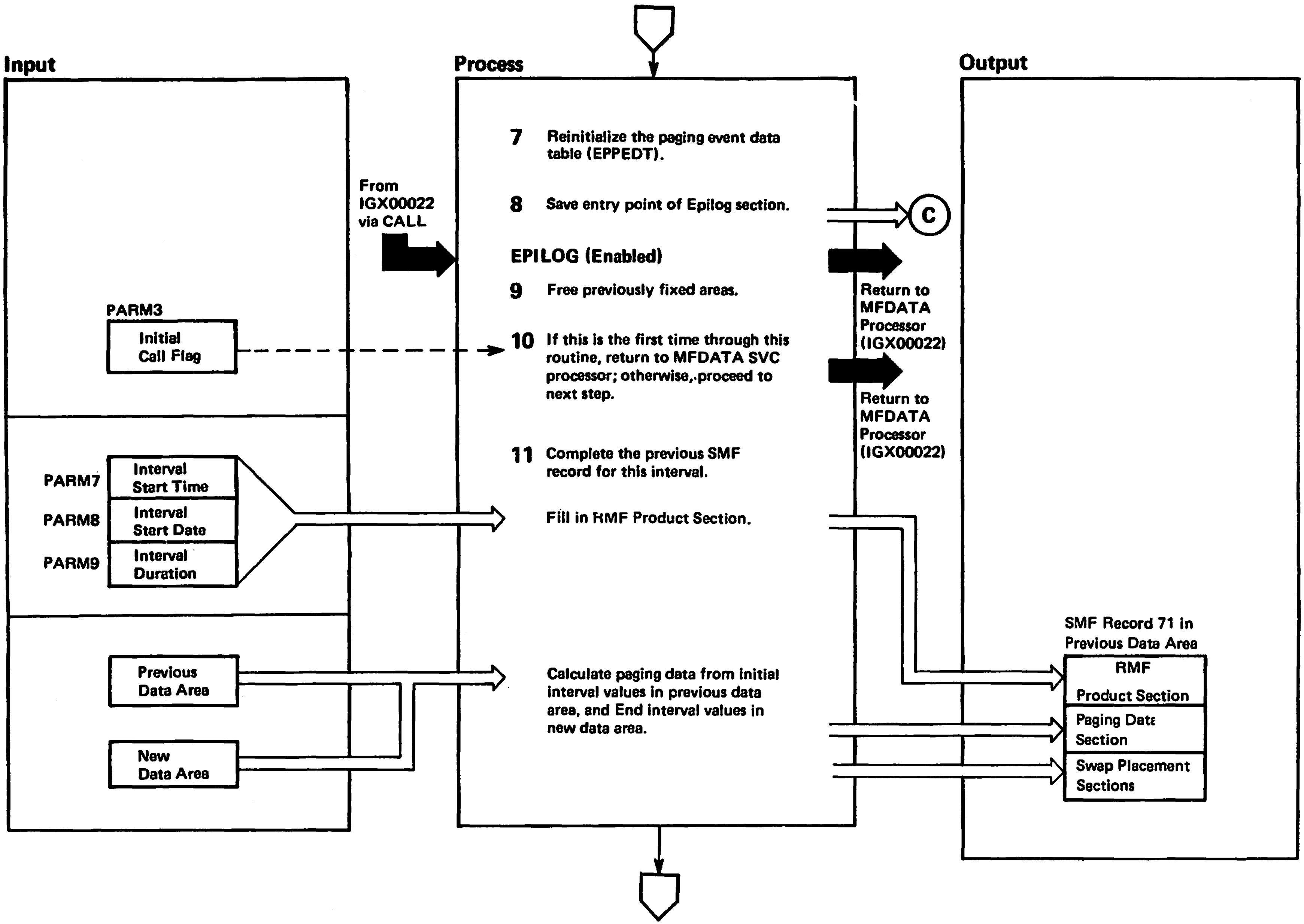


Diagram 37. Interval MG Routine for Paging (ERBMFDPP) (Part 4 of 6)

Extended Description	Module	Label
7 ERBMFDPP reinitializes the fields in the EPPEDT before sampling begins for the next interval. The table is zeroed, then fields containing minimum values are initialized to X'FF's.	ERBMFDPP	
8 ERBMFDPP saves the entry point to be used to enter its EPILOG section.	ERBMFDPP	
EPILOG		
9 If this is not the first time through step 9, ERBMFDPP uses the PGSER macro instruction with the free parameter to allow paging of previously fixed areas. The two STSGT entries for ERBMFDPP's automatic storage and for the previous data area are removed by decrementing the STRVNSGT index in the STRVT. The areas freed are those fixed in step 3. If this is the first time through step 9, no freeing takes place.	ERBMFDPP	
10 ERBMFDPP checks whether this is the initialization call from initialization mainline (MFIMAINL) and MFDATA SVC processor (IGX00022). If so, this is the start of the first interval and only initial data was saved in the SMF record in the new data area during the MOVE section. The SMF record will be used at the end of the interval. There is no SMF record in a previous data area to complete and write; thus, ERBMFDPP returns to IGX00022.	ERBMFDPP	
11 ERBMFDPP completes the SMF record in the previous data area to reflect paging data for this interval: a. The RMF Product Section is filled in with interval start time, start date and duration. b. ERBMFDPP calculates the paging data contained in wrap-around count fields by subtracting the start-of-interval values in the previous data area (saved on the last call) from the end-of-interval values in the new data area (saved on this call).	ERBMFDPP	

Diagram 37. Interval MG Routine for Paging (ERBMFDPP) (Part 5 of 6)

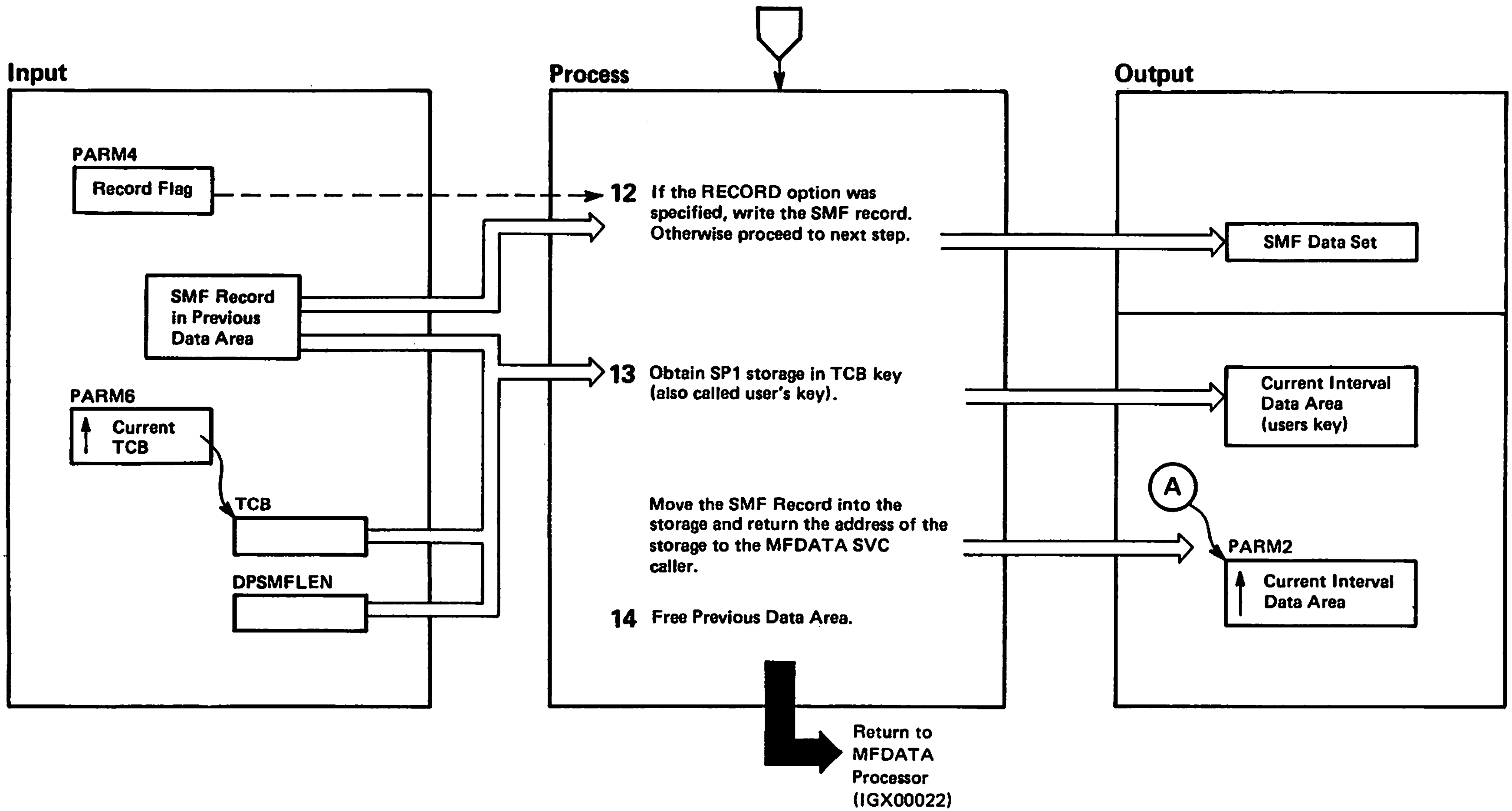


Diagram 37. Interval MG Routine for Paging (ERBMFDPP) (Part 6 of 6)

Extended Description	Module	Label
12 If RMF was started with the RECORD option, ERBMFDPP writes the SMF71HDR internal image (completed in the previous data area) to the SMF data set using the SMFWTM macro (SVC 83).	ERBMFDPP	
13 ERBMFDPP uses the GETMAIN macro instruction to obtain storage in user key (subpool 1), changes to the user key via the MODESET macro instruction, and moves the completed SMF record image to user storage. The image is copied from key 0 storage to user storage to allow the report generator module to access it in problem state/user key. *Note,* a MODESET from supervisor key to user key is done to preserve the integrity of the data during the move of the SMF record image.	ERBMFDPP	
14 ERBMFDPP uses the FREEMAIN macro instruction to free subpool 0 storage containing the SMF record completed for this interval.	ERBMFDPP	

Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 1 of 12)

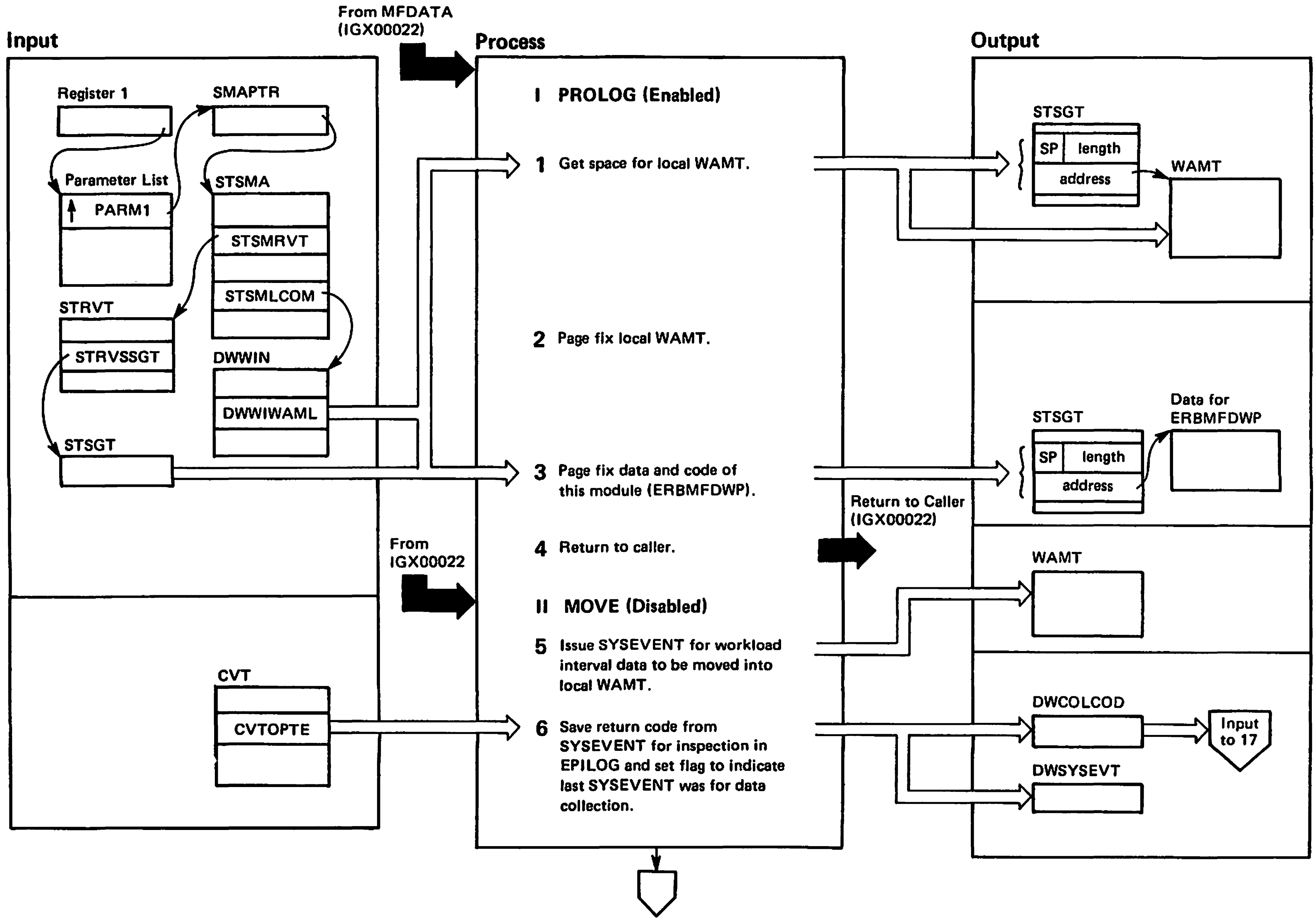
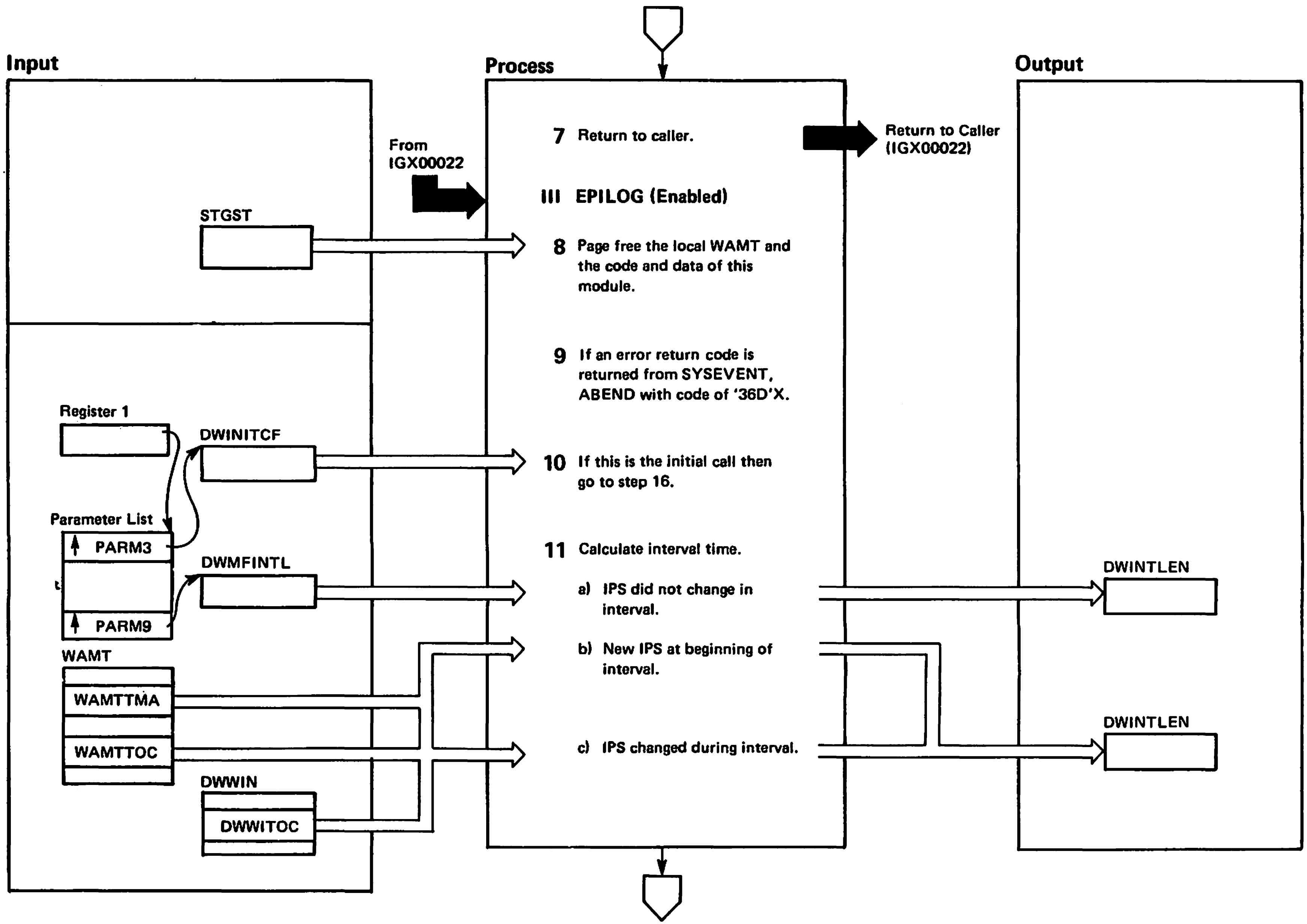


Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 2 of 12)

Extended Description	Module	Label
The interval measurement gathering routine for workload (ERBMFDWP) builds the interval image of SMF-72 records from data collected by the workload manager of the system resource manager (SRM). If required by input option selection, ERBMFDWP writes the SMF records to the SMF output data set.		
PROLOG – This portion of ERBMFDWP executes in an enabled state.		
1 An unconditional GETMAIN is performed for a local workload activity measurement table (WAMT). This WAMT storage is noted in the storage resource table (STSGT).	ERBMFDWP	
2-3 ERBMFDWP uses the PGSER macro to page fix itself, its data area and the local WAMT obtained in step 1.	ERBMFDWP	
4 ERBMFDWP uses a special return sequence which allows the caller to reenter the module at the MOVE section of ERBMFDWP in the disabled state.	ERBMFDWP	
MOVE – This portion of ERBMFDWP executes in a disabled state.		
5 ERBMFDWP issues a SYSEVENT for workload collection. This generates a branch entry to the workload manager of the SRM and causes data to be copied from the global WAMT to the local WAMT.	ERBMFDWP	
6 The return code from SYSEVENT is saved in DWCDLCOD for inspection in EPILOG processing. Also the DWSYSEVT field is set to indicate that the last SYSEVENT was issued for workload collection.		

Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 3 of 12)



7 Return to caller.

III EPILOG (Enabled)

8 Page free the local WAMT and the code and data of this module.

9 If an error return code is returned from SYSEVENT, ABEND with code of '36D'X.

10 If this is the initial call then go to step 16.

11 Calculate interval time.

a) IPS did not change in interval.

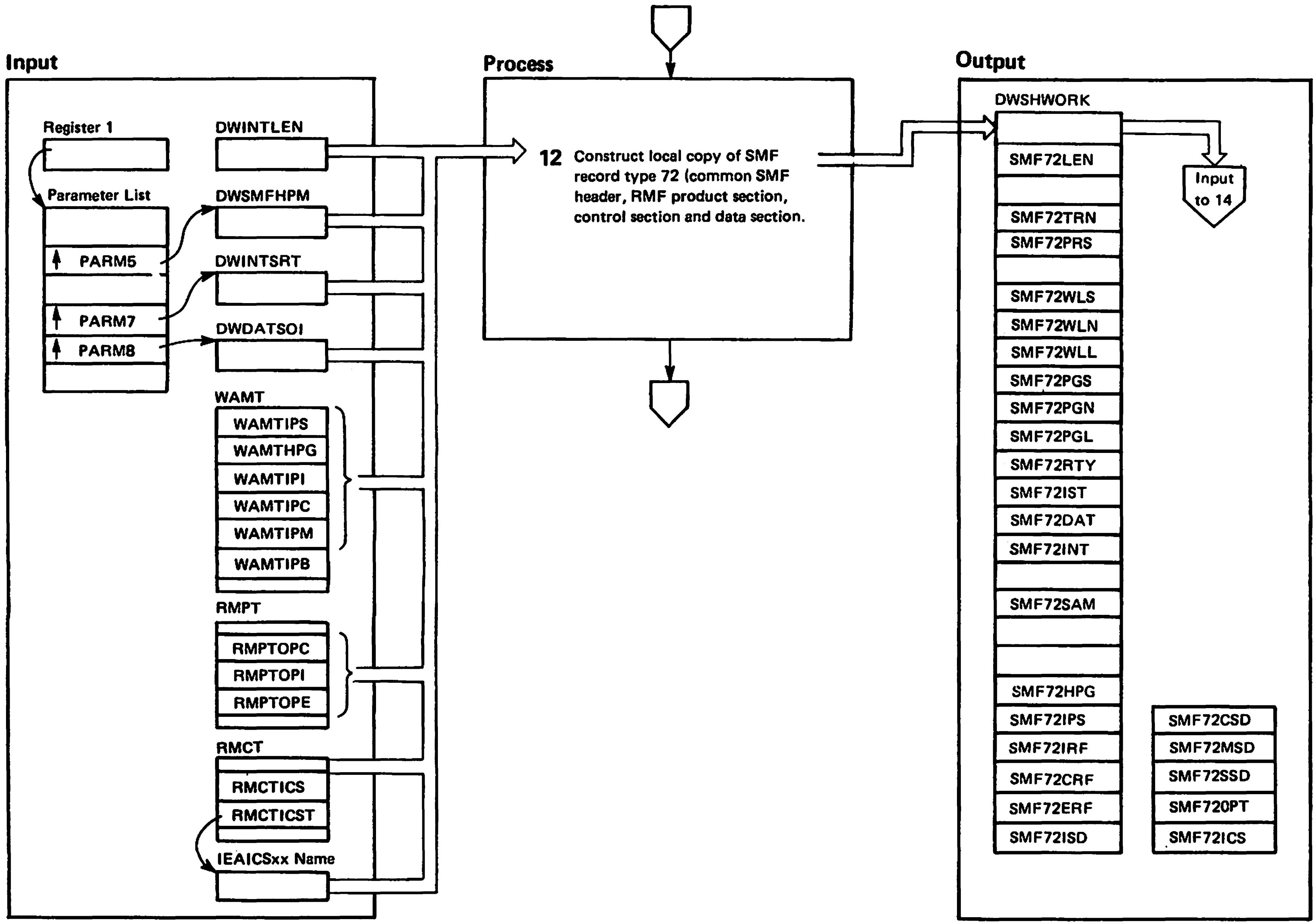
b) New IPS at beginning of interval.

c) IPS changed during interval.

Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 4 of 12)

Extended Description	Module	Label
7 ERBMFDWP uses the special return sequence to allow the caller to reenter the module at the EPILOG section of ERBMFDWP in the enabled state.	ERBMFDWP	
EPILOG – This portion of ERBMFDWP executes in an enabled state.		
8 ERBMFDWP uses the PGSER macro to page-free those areas which were fixed in steps 2 and 3.	ERBMFDWP	
9 If the return code from SYSEVENT indicates an error (RC ≠ 0 and RC ≠ 8), ERBMFDWP issues an ABEND with a code of '36D'X. The reason code is the SYSEVENT return code.	ERBMFDWP	
10 If this is the initial call to ERBMFDWP, then go to step 16; otherwise, continue with step 11.	ERBMFDWP	
11 ERBMFDWP calculates the interval time and places it in the field DWINTLEN for use in constructing SMF records.	ERBMFDWP	

Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 5 of 12)



Extended Description	Module	Label
12 ERBMFDWP builds a local copy of a type 72 SMF record. This copy contains SMF data that is common to all SMF records (the SMF common header, and RMF product section), as well as workload control section and workload data section. The offset/length/number triplets are calculated for all sections and inserted in the header (product section) and extended header.	ERBMFDWP	

The entire length of the SMF record is calculated as follows:

(offset to the control section)+(length of workload control section)*(number of control sections)+(length of workload data section)*(number of data sections)

The number of workload data sections depends on the number of performance group periods specified in the installation performance specifications (IPS).

Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 7 of 12)

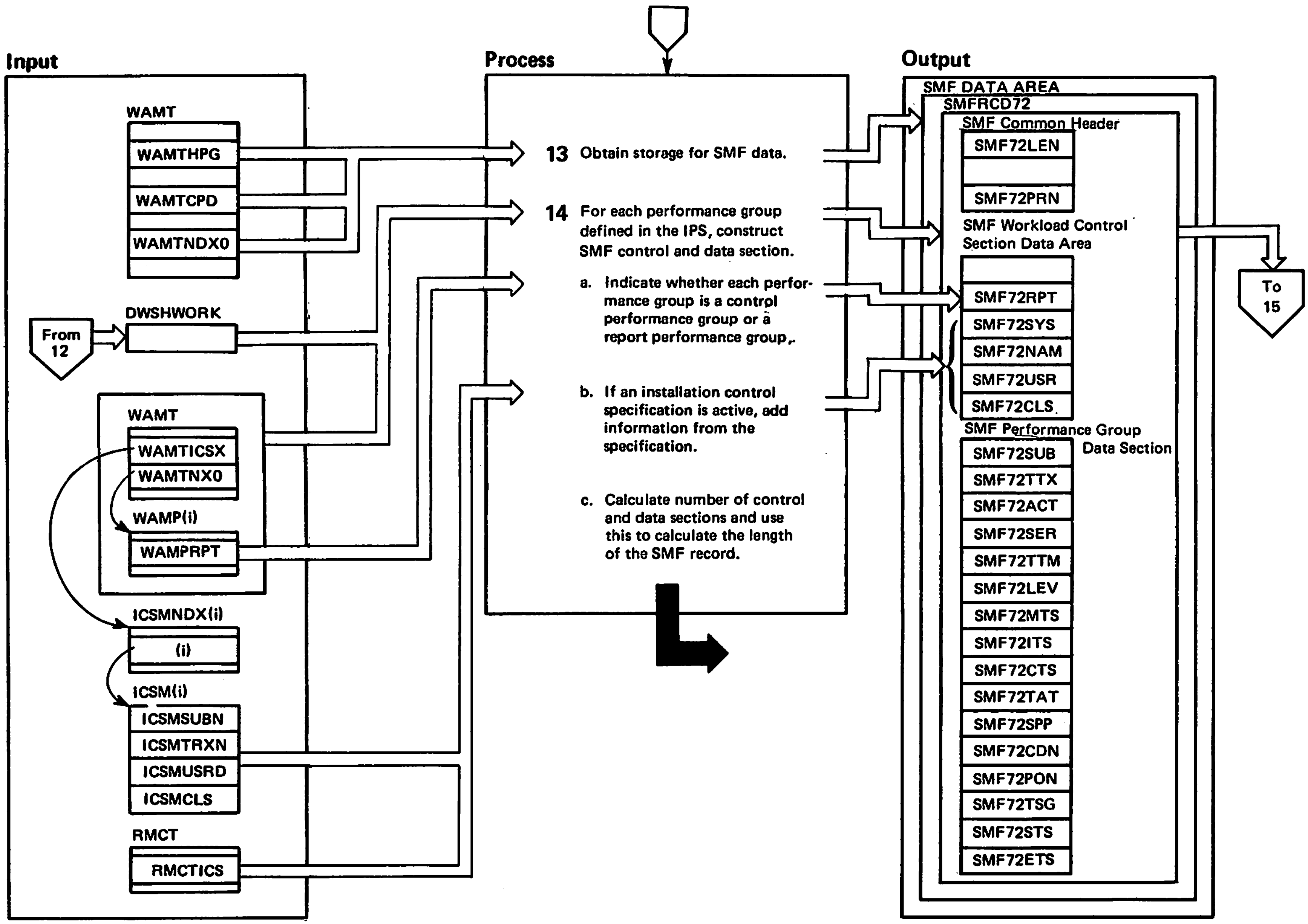
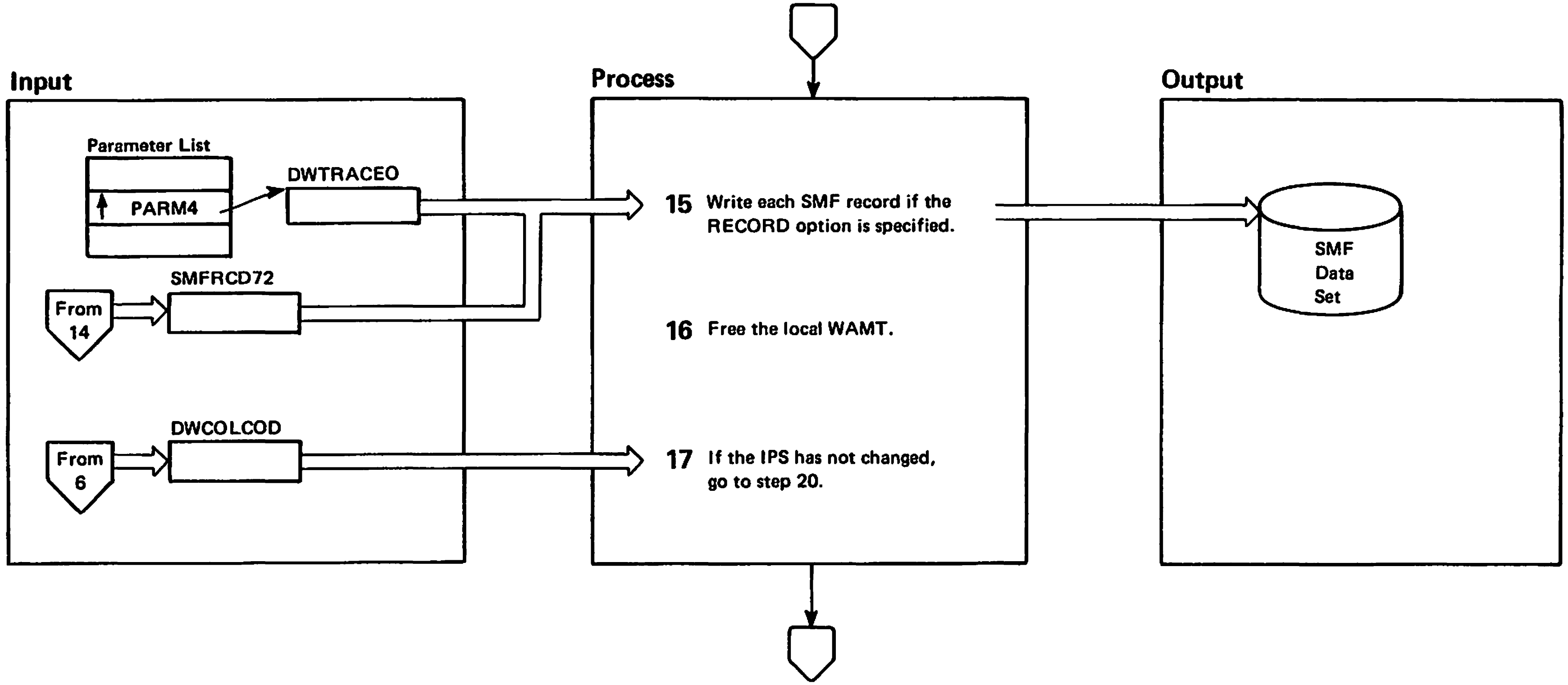


Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 8 of 12)

Extended Description	Module	Label
13 ERBMFDWP uses the GETMAIN macro to obtain storage for the SMF data area. This area is used to build a table of pointers to the SMF record images and also contains the SMF record images. The amount of storage needed for this data area is: (length of subpool id and length prefix) + (length of performance group prefix) + (length of SMF record pointer table) + (number of performance groups) * (length of SMF record +length of RMF product section+length of workload control section)+data section	ERBMFDWP	
14 ERBMFDWP constructs an SMF record image for each performance group defined in the IPS. Common data in the SMF records is obtained from DWSHWORK; the data for each performance group period comes from the local WAMT. a. Each performance group is checked to see if it is in a report-only class (WAMPRT = ON). If so, '**' is used in place of the objective, domain, and time slice group data. b. If an installation control specification is active (RMCTICS = ON), include the control information in the SMF record. c. Calculate the number of control and data sections, set up the offset/length/number triplets on the extended header and calculate the total length of the SMF record.	ERBMFDWP	

Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 9 of 12)



Extended Description	Module	Label
15 If the RECORD option is selected, ERBMFDWP writes an SMF record, constructed in step 14, to the SMF data set for each performance group that was active during the interval. The SMFWTM macro is used to write the record.	ERBMFDWP	
16 ERBMFDWP uses the FREEMAIN macro to free the storage of the local WAMT.	ERBMFDWP	MFFREWAM
17 If the IPS has not changed, go to step 20. Otherwise continue with step 18.	ERBMFDWP	

Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 11 of 12)

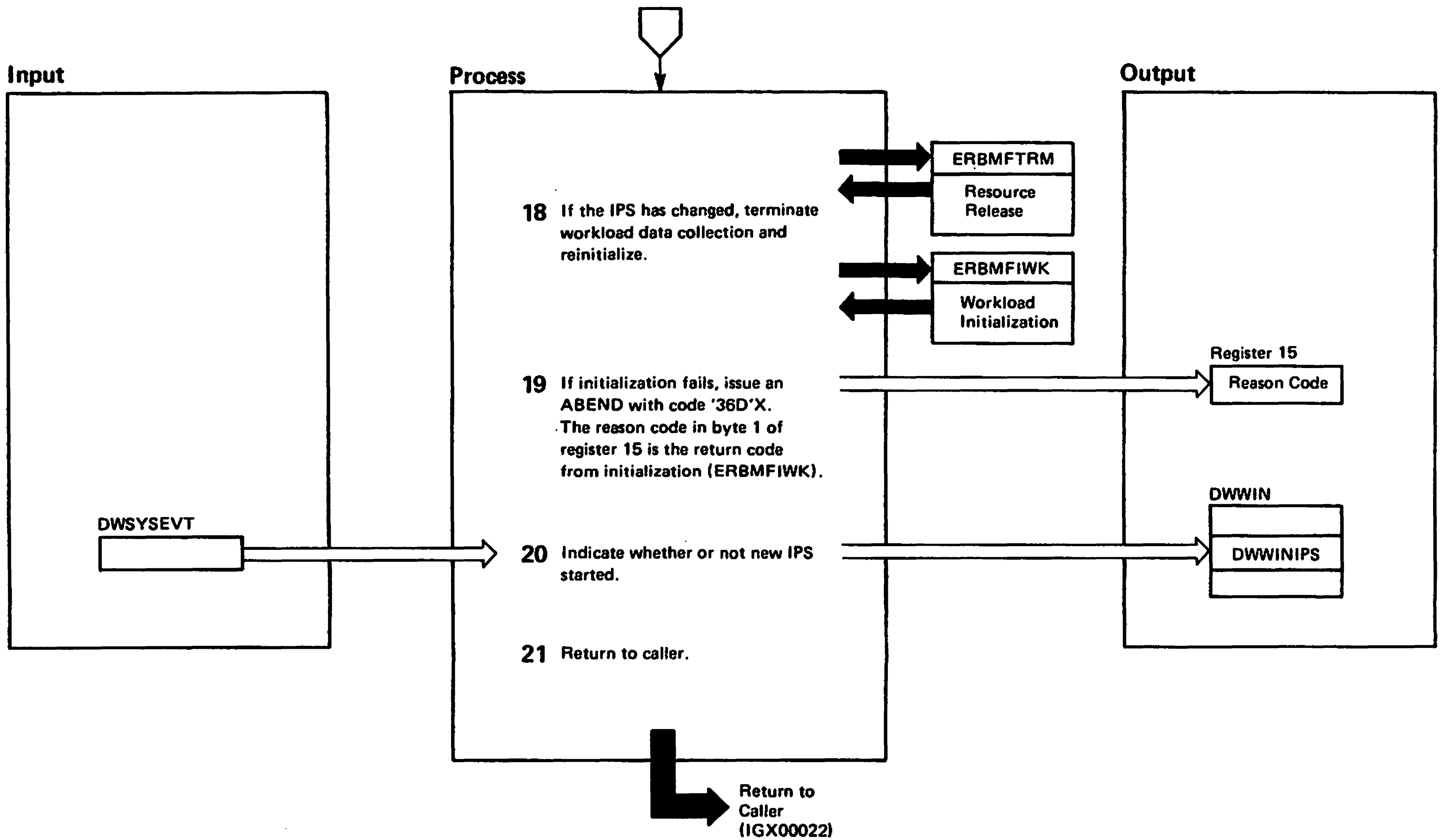


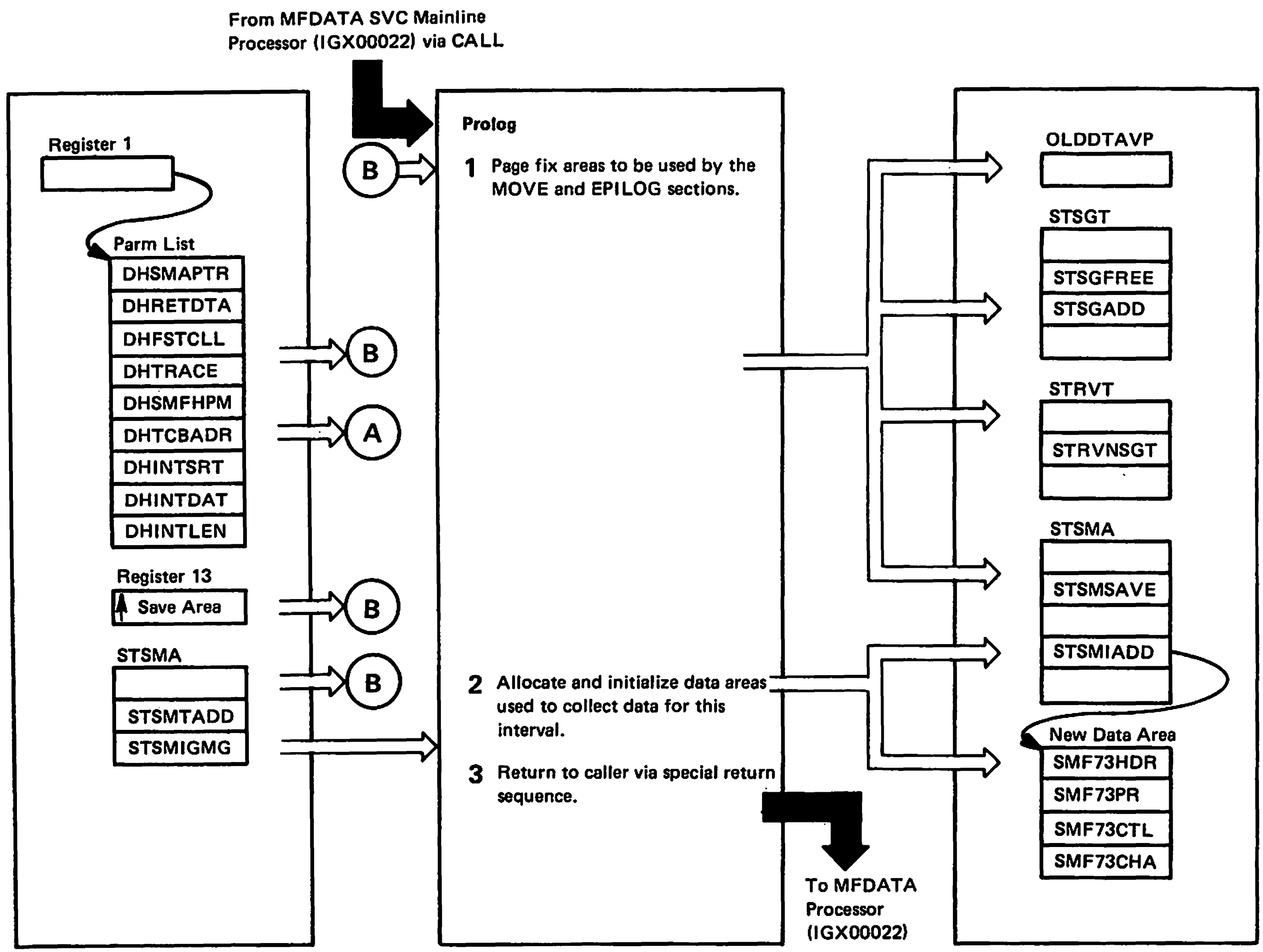
Diagram 38. Interval MG Routine for Workload (ERBMFDWP) (Part 12 of 12)

Extended Description	Module	Label
18 If the IPS has changed in the interval, ERBMFDWP issues a SYSEVENT macro to terminate workload activity data collection. It then calls ERBMFTRM to release resources and ERBMFIWK to re-initialize the collection of workload activity data.	ERBMFDWP ERBMFTRM ERBMFIWK	
19 If the reinitialization in step 18 is not successful, ERBMFDWP issues an ABEND with a code of '36D'X. The reason code in byte 1 of register 15 is the return code from the initialization module ERBMFIWK.	ERBMFDWP	
20 The DWWINIPS field in the workload interval table (DWWIN) is set to indicate whether or not a new IPS was started in the interval.	ERBMFDWP	

Error Processing:

No specific check is made for errors except for the return from the SYSEVENT in steps 9 and 19.

Diagram 39. Interval MG Routine for Channel Paths (ERBMFDHP) (Part 1 of 6)



Extended Description	Module	Label

The interval measurement gathering routine for channel paths (ERBMFDHP) receives control from the MFDATA SVC mainline processor at the end of each interval if channel path activity reports are required. ERBMFDHP obtains and formats the collected data and records the data on the SMF data set (via the SMFWTM macro instruction) if RECORD is specified as an input option.

Prolog

1 On entry, the address of a standard save area is stored into STSMSAVE field of the supervisor measurement area (STSMA). The address is later passed as a parameter on subsequent calls by IGX00022 when it reenters at the move and epilog sections. ERBMFDHP does the following:

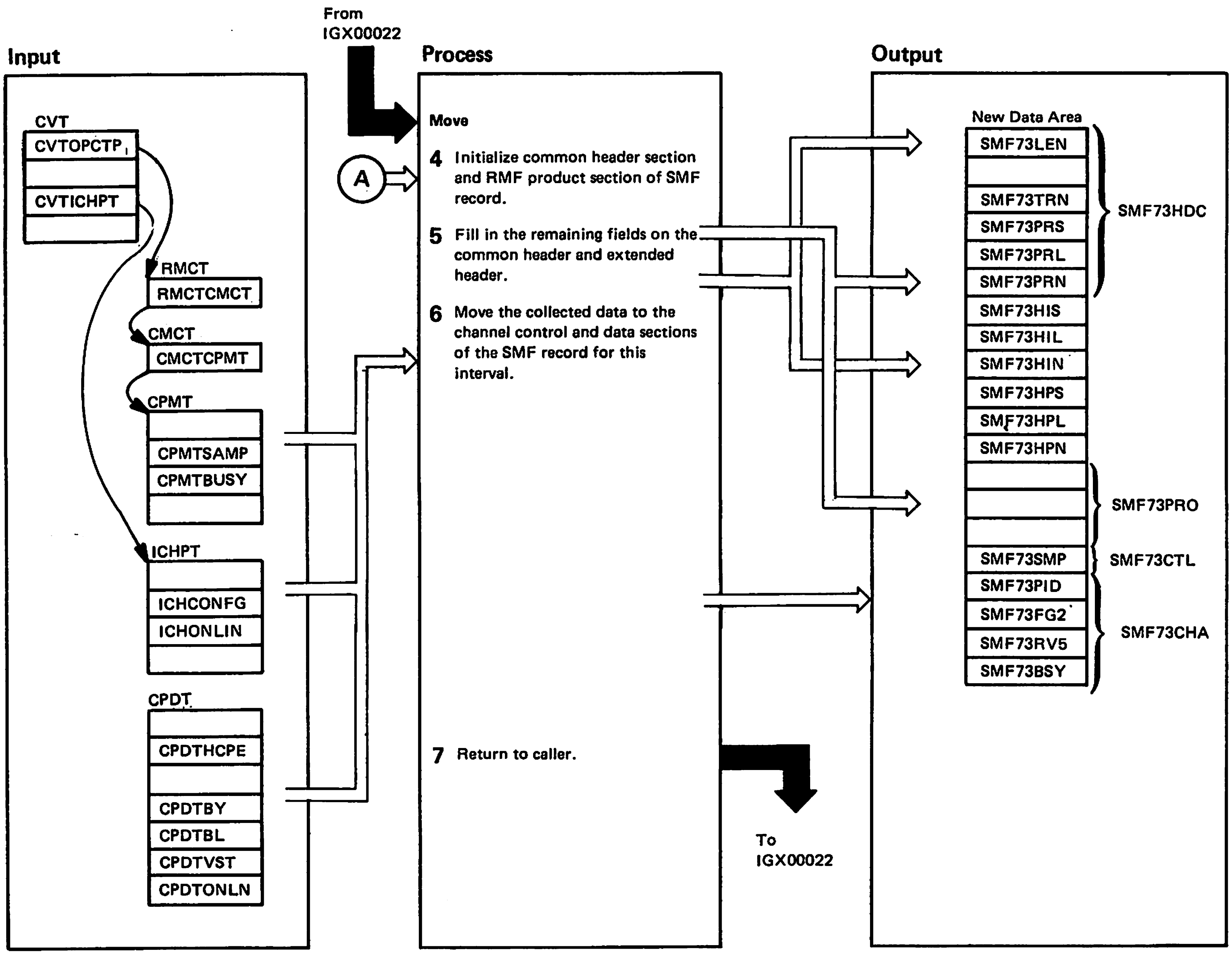
- Saves the previous data area address
- Issues the PGSER macro instruction to fix static and automatic data
- Issues the PGSER macro to fix its own code

2 Issue the GETMAIN macro instruction in key 0 to obtain the new required storage area from subpool 0 for the SMF record. Data for the next interval (the one just starting) will be collected in this storage. The address of the acquired storage is stored in the STSMIADD field of the STSMA.

- ERBMFDHP also stores the subpool number and the length of the storage area obtained into the first word of the area.
- Issues the PGSER macro instruction to fix the area. Page fixing is done to prohibit page fault interrupts during the disabled move section.

3 Saves entry point for MFDATA SVC mainline processor (IGX00022), for use in returning to the move part of ERBMFDHP.

Diagram 39. Interval MG Routine for Channel Paths (ERBMFDHP) (Part 3 of 6)



Extended Description	Module	Label

Move

4 Partially initialize the SMF record type image in storage with data provided by IGX00022: common header section (SMF73HDC) and RMF products section (SMF73PRO).

5 Complete the SMF header section image in storage

- Offset to RMF product section=length of header section
- Offset to the channel control section=offset to the RMF product section + length of the RMF product section
- Offset to the channel data section=offset to the channel control section + length of the channel control section
- Length of the entire SMF record=offset to the channel data section + length of one channel data section array entry*number of channel path identifiers in the configuration.

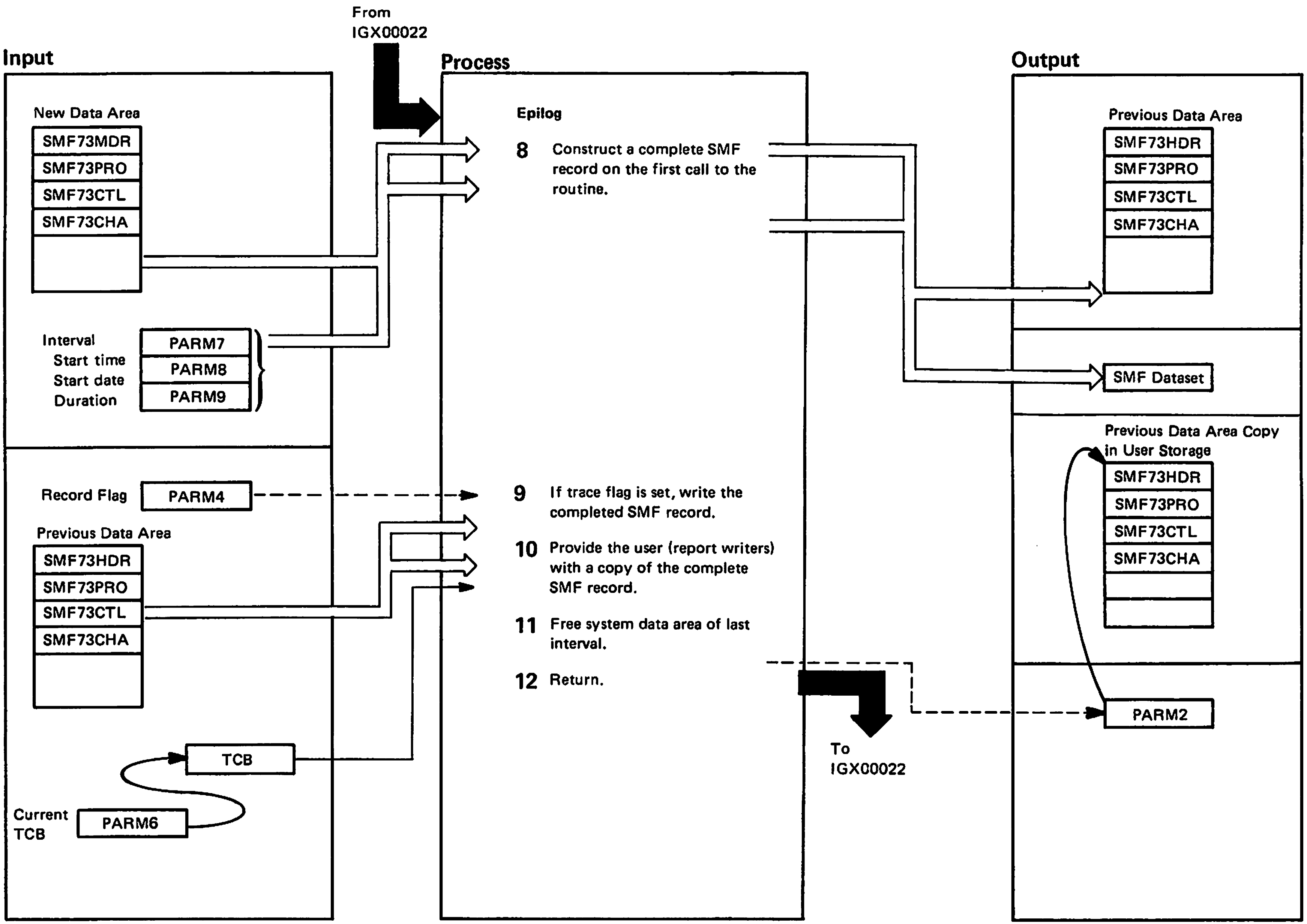
6 Move the collected data to the SMF record image in storage. Use the installed channel path table to determine which channel paths are installed. Channel paths that are not installed are skipped.

For each installed channel path, the channel path data table (CPDT) and channel path measurement table (CPMT) are used to collect the data:

- type of channel
- current status (online/offline)
- past history of channel path status
- busy count

7 Save the entry point for returning to the Epilog segment.

Diagram 39. Interval MG Routine for Channel Paths (ERBMFDHP) (Part 5 of 6)



Extended Description	Module	Label

Epilog

8 Gather the initial measurement data values on the first call. These values are required at the end of the measurement interval to calculate data for the interval. Processing ends here on the initial call. There is an SMF73CHA entry for each channel path installed in the system. Calculate the busy value by subtracting the busy value for the last interval from that for this interval. Calculate the sample value by subtracting the sample value for the last interval from that for this interval. Copy all other values gathered in this interval to the output area.

9 Copy the internal image of the SMF record to the SMF data set by use of the SMFEWTM macro instruction.

10 ERBMFDHP issues the GETMAIN macro instruction to obtain storage in user key (subpool 1), changes to user key via MODESET macro instruction, and moves the completed SMF record image to the user storage. The image is copied from key 0 storage to the user storage to allow the report generator modules to access it in problem state/user key. The MODESET from supervisor key to user key is done to preserve the integrity of the data during the move of the SMF record image.

11 Issue a FREEMAIN macro instruction to release the subpool 0 storage used for the internal image of the SMF record.

12 Return to caller IGX00022.

Diagram 40. Interval MG Routine for Devices (ERBMFDDP) (Part 1 of 8)

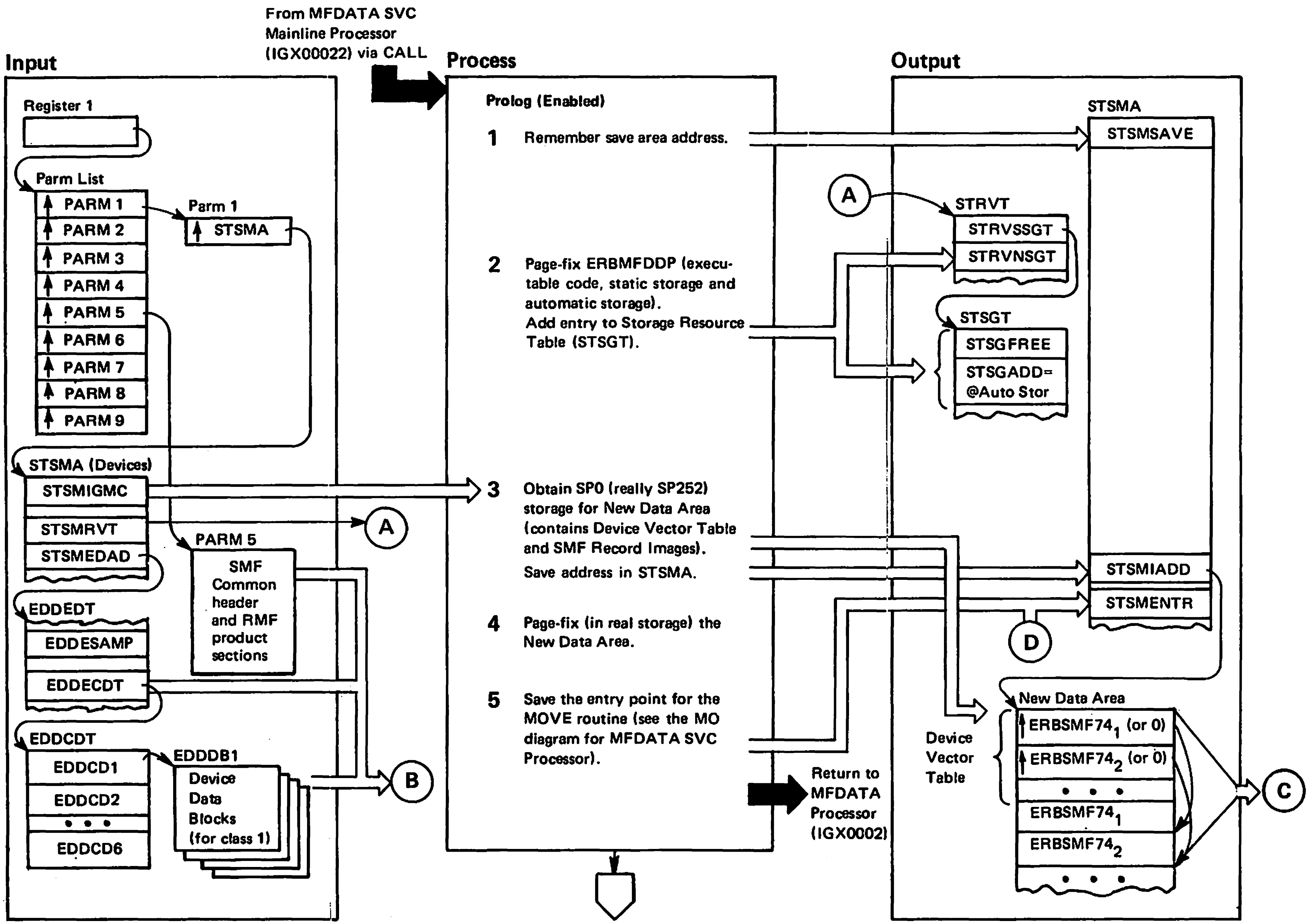
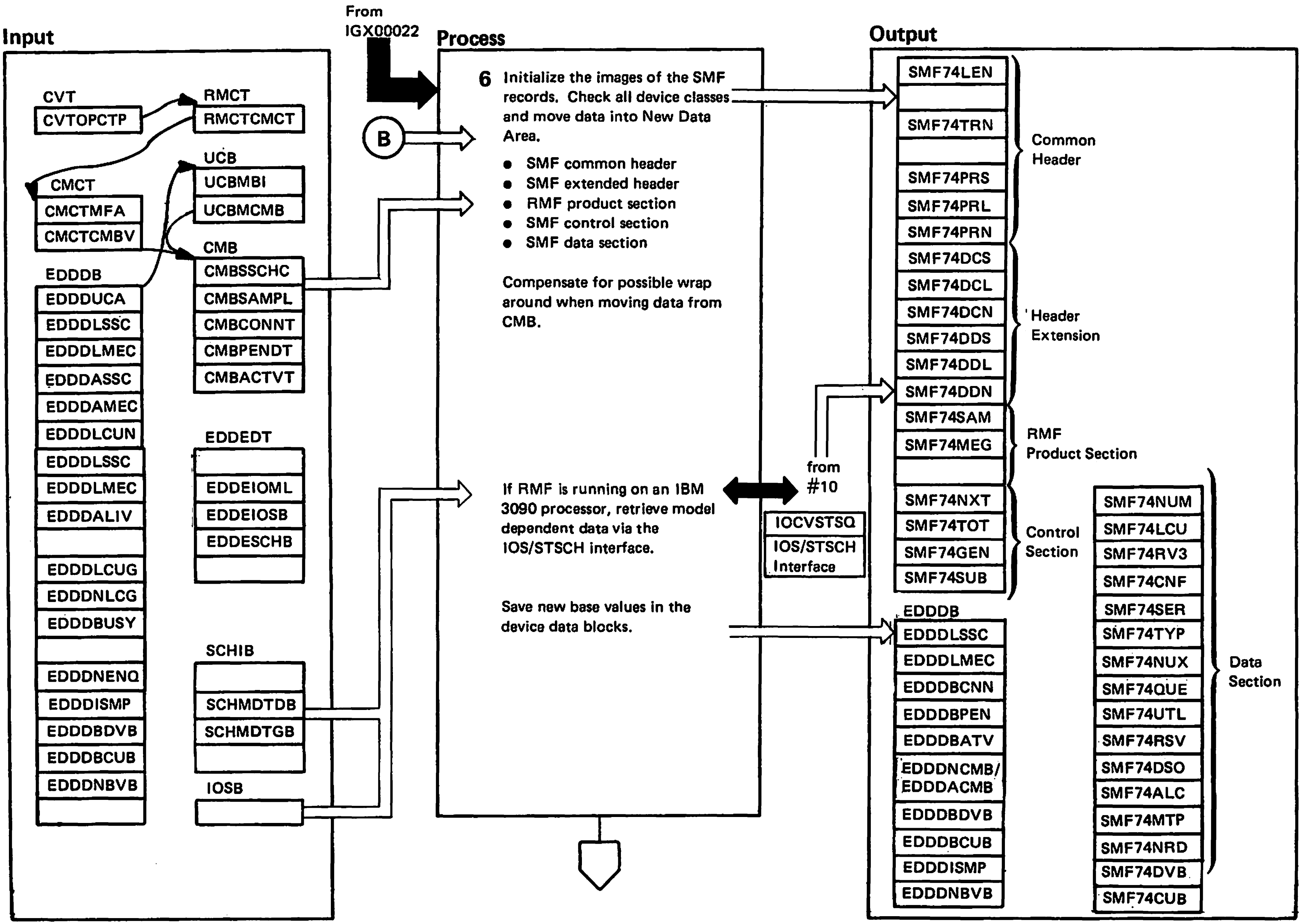


Diagram 40. Interval MG Routine for Devices (ERBMFDDP) (Part 2 of 8)

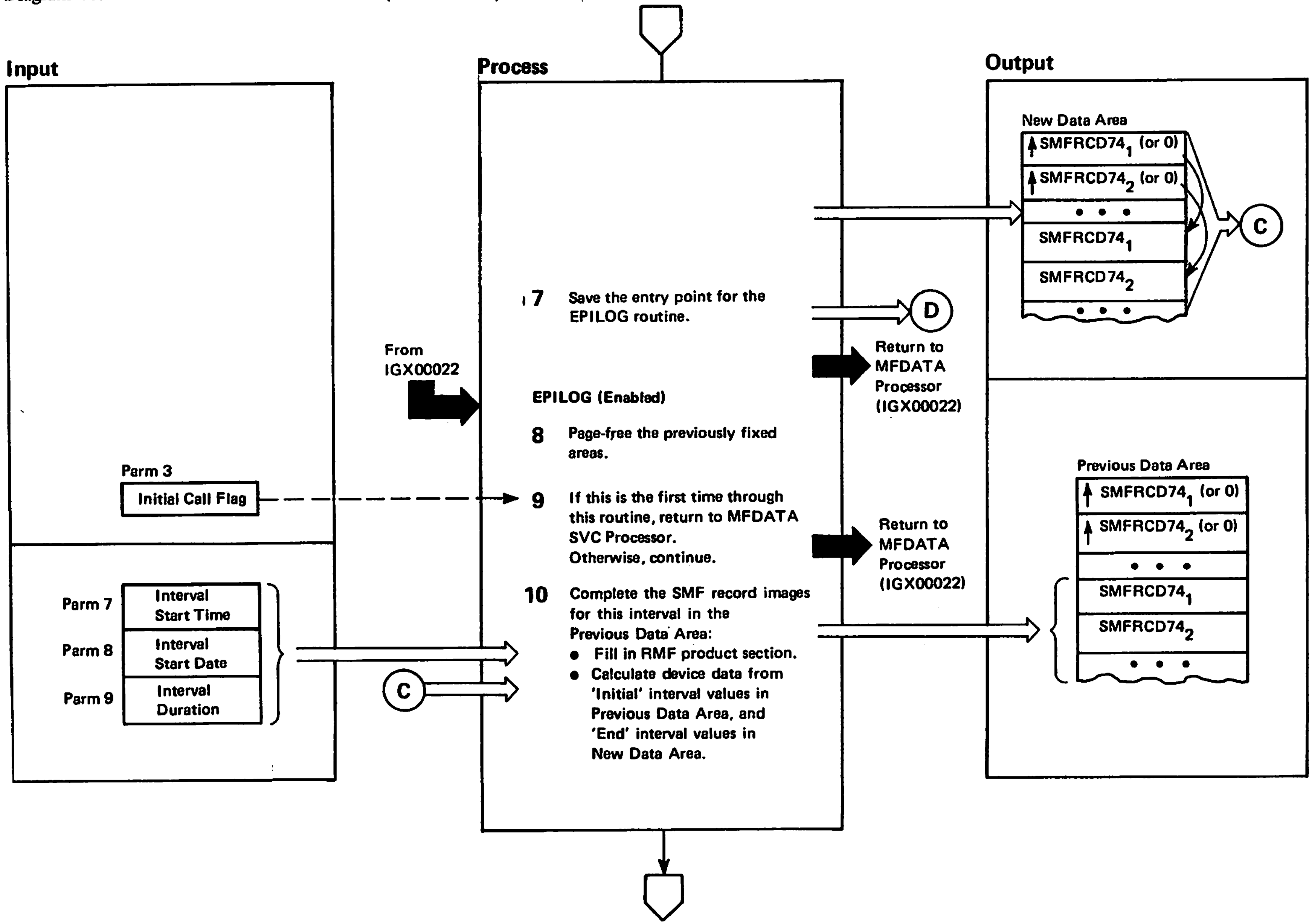
Extended Description	Module	Label
The Interval Routine for Devices (ERBMFDDP) receives control from the MFDATA SVC Mainline Processor (IGX00022) at the end of each interval if any device reports are required. ERBMFDDP builds the internal image of one or more device data SMF records (ERBSMF74; one to nine records for each class of device reports requested) from data collected in event control blocks (EDDDBs). Fields from associated unit control blocks (UCBs) and channel measurement blocks (CMBs) are also used to complete the SMF record. If requested in the input options, ERBMFDDP copies the internal record images to the SMF data set (via the SMFWTM macro instruction).		
Prolog (Enabled)		
1 On entry, the address of a standard save area is stored into the STSMSAVE field of the Supervisor Measurement Area (STSMA). The address is later passed as a parameter on subsequent calls by IGX00022 when it reenters at the MOVE and EPILOG sections.	ERBMFDDP	
2 ERBMFDDP uses the PGSER macro instruction to inhibit paging of the ERBMFDDP routine, its local storage and its automatic storage. An entry is made in the Storage Resource Table (STSGT) for the ERBMFDDP's automatic storage. (Subpool and length are in the STSGFREE field and the address is in the STSGADD field). Reset the index in the STRVNSGT field of the Resource Vector Table (STRVT) to the next available STSGT entry.	ERBMFDDP	
Note: Page fixing is done to prohibit page fault interrupts during the disabled MOVE section.		
3 The GETMAIN macro instruction is used to obtain SP0 (really SP252 since requested in Supervisor state) storage for a New SMF record data area. Start of interval data for the next interval will be collected in this storage. The address is saved in the STSMIADD field of the STSMA after the Previous Data Area's address is saved.	ERBMFDDP	
4 Store the subpool and length of the storage obtained into the first word of the area. Use the PGSER macro instruction to fix the new data area. Calculate the maximum number of device data sub records that will fit in one SMF record.	ERBMFDDP	
5 Save the entry point, as described in the M.O. diagram, MFDATA SVC Mainline Processor (IGX00022), for use in returning to the Move part of ERBMFDDP.	ERBMFDDP	DDMOVE

Diagram 40. Interval MG Routine for Devices (ERBMFDDP) (Part 3 of 8)



Extended Description	Module	Label
6 Initialize the images of the SMF records. Check all device classes (one class is associated with each EDDCD), and move data from the channel measurement blocks (CMBs) and the device event data table (EDDEDT) into the SMF record image corresponding to that device class. If no device exists for this class or if no measurements have been requested for a class, set the pointer for the SMF record images to zero.		
Compensate for possible wrap around when moving data from the CMB.		
If RMF is running on an IBM 3090 processor, invokes the IOS/STSCH interface to obtain the new model-dependent data provided by the hardware. Calculates the difference between the old and new value and moves the value into the SMF record, then clears the model-dependent data flag to indicate that base values are available. If the STSCH interface fails, increments the counter for unsuccessful samples (EDDDISMP), moves the invalid sample count into the SMF record, and indicates that no base values are available for the next interval.		

Diagram 40. Interval MG Routine for Devices (ERBMFDDP) (Part 5 of 8)



Extended Description	Module	Label
7 Save entry for returning to Epilog segment.	ERBMFDDP	
Epilog (Enabled)		
8 ERBMFDDP uses the PGSER macro instruction to allow paging of previously fixed areas. The STSGT entry for ERBMFDDP's automatic storage is removed by decreasing the STRVNSGT index in the STRVT.		
9 On the initializing call from the MFDATA SVC, the interval driven MG routines obtain an initial set of values for wrap-around measurements to be used at the end of the first interval when calculating data for that interval. Processing ends here on that call.	ERBMFDDP	DDEPILOG

Extended Description	Module	Label
10 For each device class for which a device exists and for which measurements are required, place the data for the interval just ended into the record in previous data area obtained during the previous interval, overlaying previous data where necessary. For each SMF record 74 device data section, which exists for a device whether it appears online at all during the interval, the determination is made whether or not to keep it. If no device measurements are associated with the device data section (SMF74B), the record is eliminated and the other records compressed. All the SMF74 records retained are sorted into order of ascending device numbers within a logical control unit. If no LCU information is available, the records are sorted into order of ascending device numbers.	ERBMFDDP	

Diagram 40. Interval MG Routine for Devices (ERBMFDDP) (Part 7 of 8)

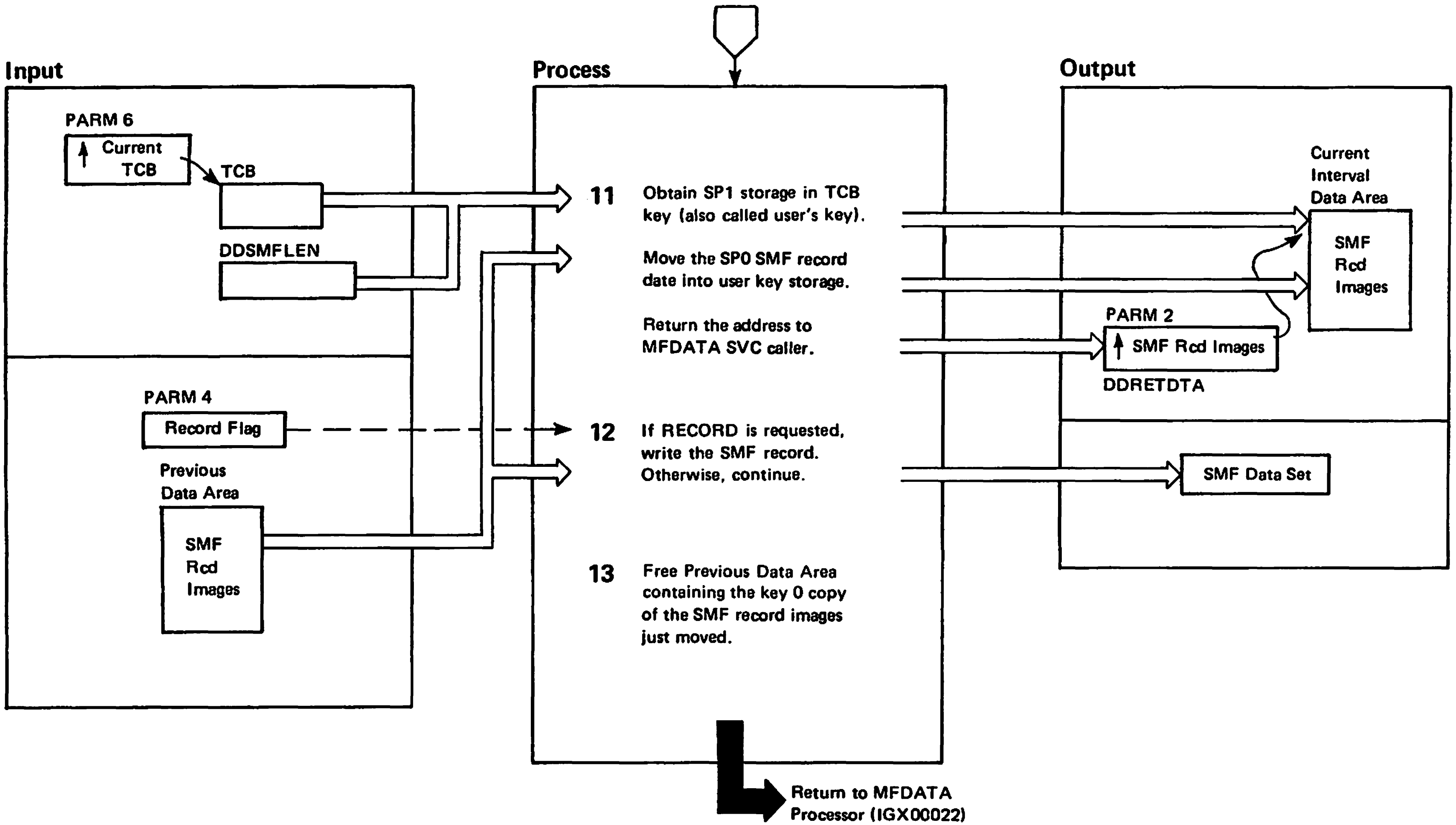


Diagram 40. Interval MG Routine for Devices (ERBMFDDP) (Part 8 of 8)

Extended Description	Module	Label
11 Use the GETMAIN macro instruction to obtain the required storage in the user key. This area is used to return SMF data records. Use the MODESET macro instruction to change to the user key. Move the completed SMF record images to the user key storage. Build one to nine records as needed in the user key storage from the large record in supervisor key storage. The data is copied from key 0 storage to user storage to allow the report generator module to access it in problem state/user key. Note, a MODESET from supervisor key to user key is done for integrity during the move of the SMF record images.	ERBMFDDP	
12 Use the SMFWTM macro instruction to copy the internal images to the SMF data set, if requested.	ERBMFDDP	
13 Use the FREEMAIN macro instruction to release the SP0 (really SP252) storage obtained for the internal images of SMF records.	ERBMFDDP	

Diagram 41. Interval MG Routine for Trace (ERBMFDTP) (Part 1 of 6)

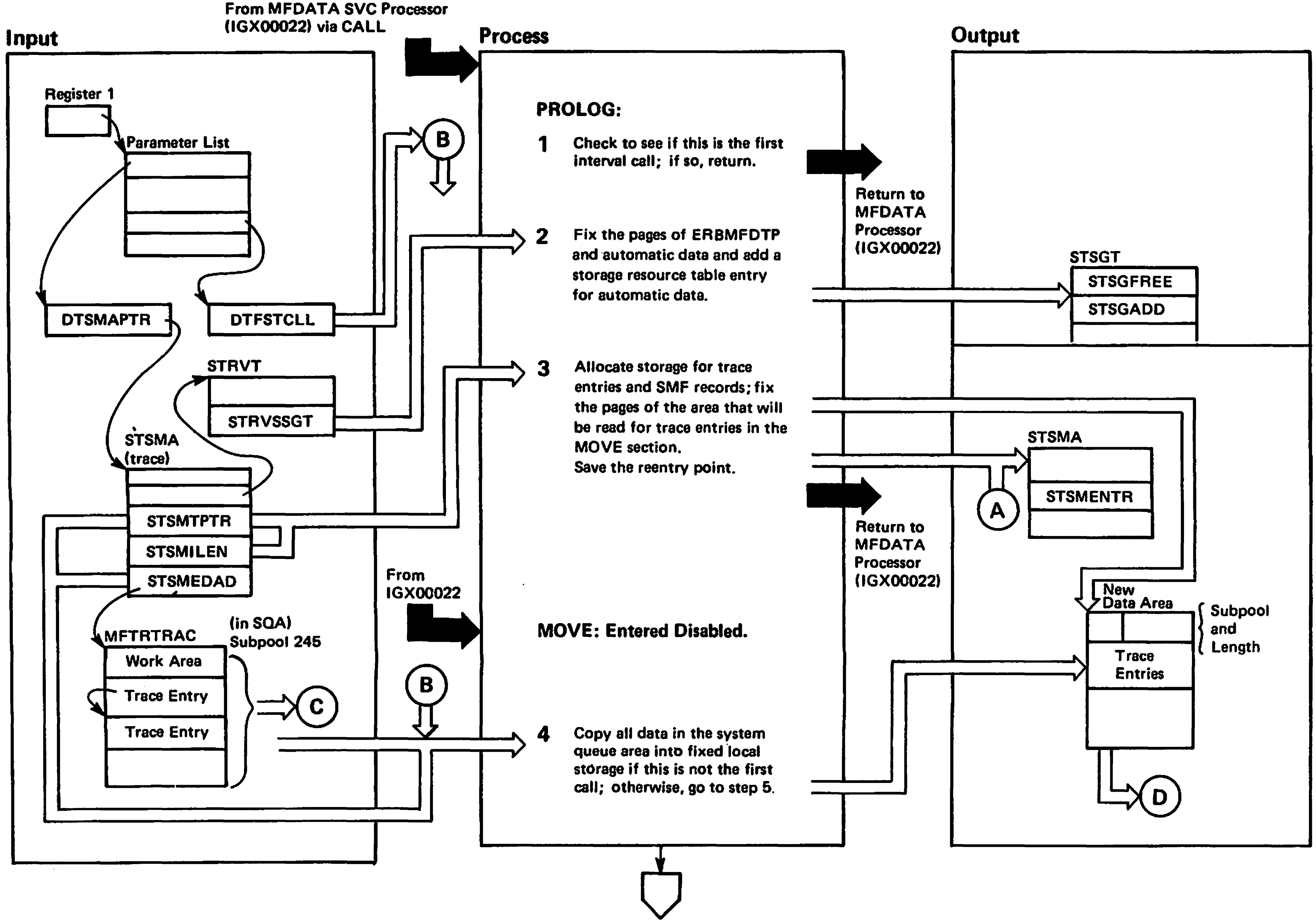


Diagram 41. Interval MG Routine for Trace (ERBMFDTP) (Part 2 of 6)

Extended Description	Module	Label
ERBMFDTP receives control from the MFDATA SVC processor if trace is active. ERBMFDTP changes the trace entries in the system queue area to type 76 SMF records, resets all counters and accumulators in the system queue area so that trace sampling can be restarted on the new interval, and writes the SMF records to SMF dataset if requested.	ERBMFDTP	
1 ERBMFDTP returns using register 14 if this is the first call (DTFSTCLL='1').	ERBMFDTP	
2 The PGSER macro is used to page fix ERBMFDTP and construct a storage resource table entry for the automatic data. For each storage resource address ERBMFDTP stores the subpool, length and address of the added storage in the STSGT.	ERBMFDTP	
3 Using the length (STSMILEN) in the STSMA provided by ERBMFITR, a GETMAIN is issued for all local storage needed for a local copy of the trace entries and the SMF record to be built. The PGSER macro is used to fix only the storage required for trace entries during the MOVE section. A control word is inserted containing the length and the subpool at top of the area. The reentry point (i.e., the MOVE section address) is saved and return is made to MFDATA using register 14.	ERBMFDTP	DTMOVE
4 If this is not the first call, ERBMFDTP moves the entire block of trace entries from the system queue area into the fixed local store. The field, STSMTPTR, in the STSMA contains the total trace data length. This length was set by ERBMFITR.	ERBMFDTP	

Diagram 41. Interval MG Routine for Trace (ERBMFDTP) (Part 3 of 6)

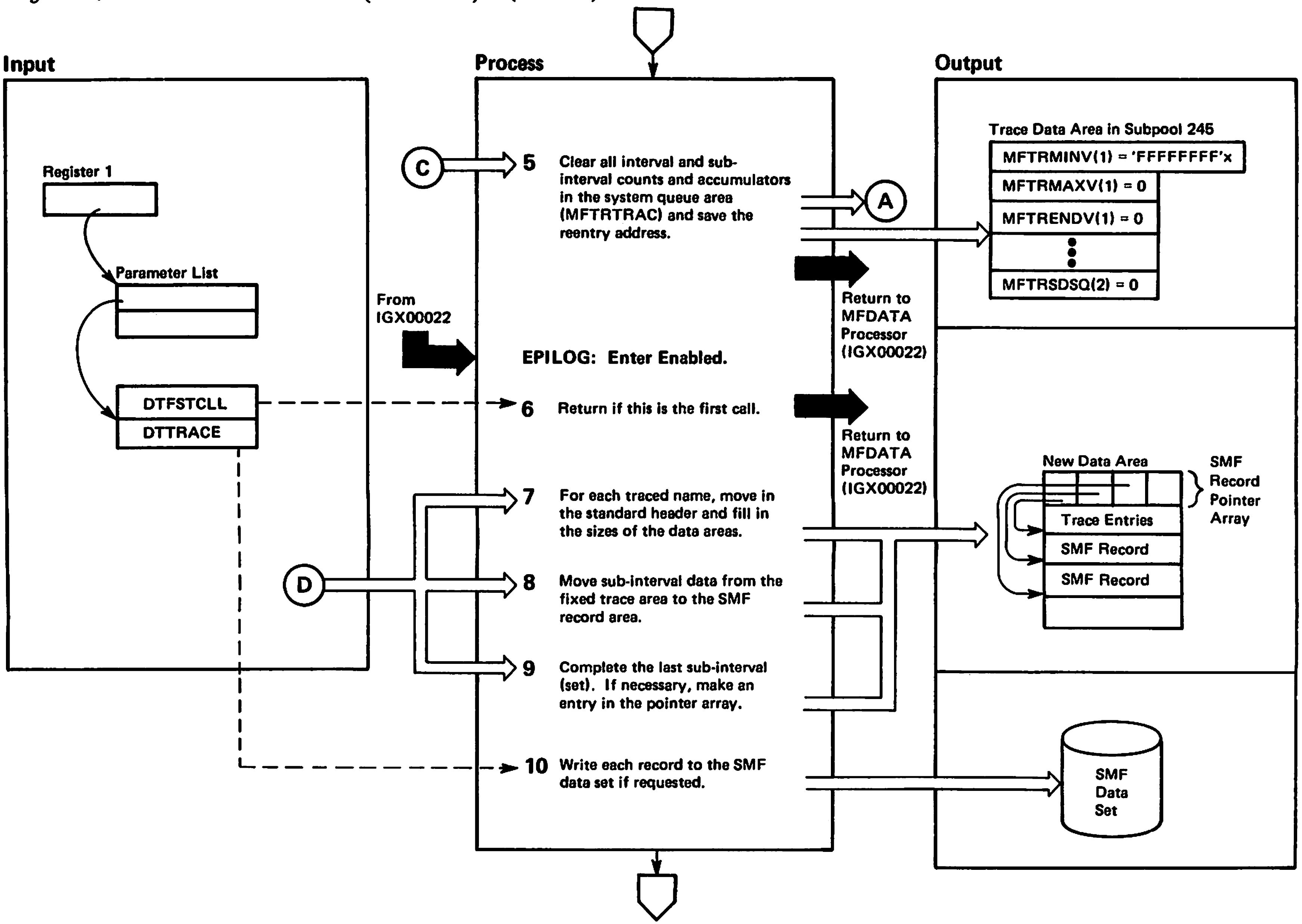


Diagram 41. Interval MG Routine for Trace (ERBMFDTP) (Part 4 of 6)

Extended Description	Module	Label
5 Using pointer, STSMEDAD, in the STSMA, ERBMFDTP processes each trace entry in the system queue area (MFTRTRAC) and clears all counts; reinitializes all accumulators for a new interval, saves the entry address, and returns.	ERBMFDTP	DTEPILOG
6 If this is the first call ERBMFDTP returns; otherwise, steps 7-10 are repeated for all traced names.	ERBMFDTP	
7 ERBMFDTP moves in the standard SMF header to build record parts SMF76HDR and SMF76PRO. ERBMFDTP fills in time, date, record type, number of sub-intervals, and the offset/length/number triplets in the header extension.	ERBMFDTP	
9 If there is still data in the sub-interval accumulators, a 'last set' must be built to include it. If the MFTRCNT(1) field is 0, a new set is put on the end of the SMF76 record using the same method as trace sampling, ERBMFETR. The address of the new SMF76 record is added to a list formed at the top of the fixed local trace entries. This array of pointers will later be used by the report generator ERBMFRTR to locate the records.	ERBMFDTP	
10 If the RECORD option is requested (DTTRACE='1'), ERBMFDTP uses the SMFWTM macro to write the SMF76 record to the SMF data set.	ERBMFDTP	

Diagram 41. Interval MG Routine for Trace (ERBMFDTP) (Part 5 of 6)

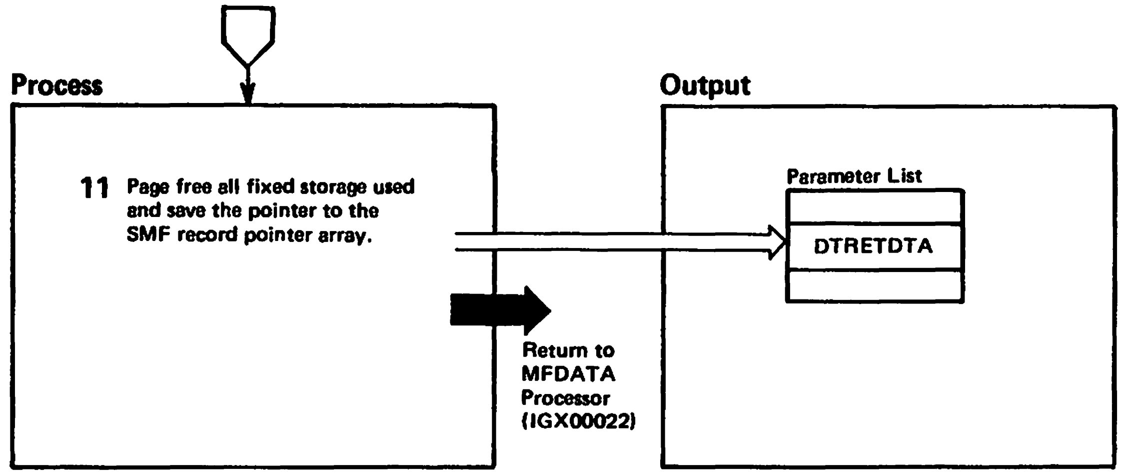
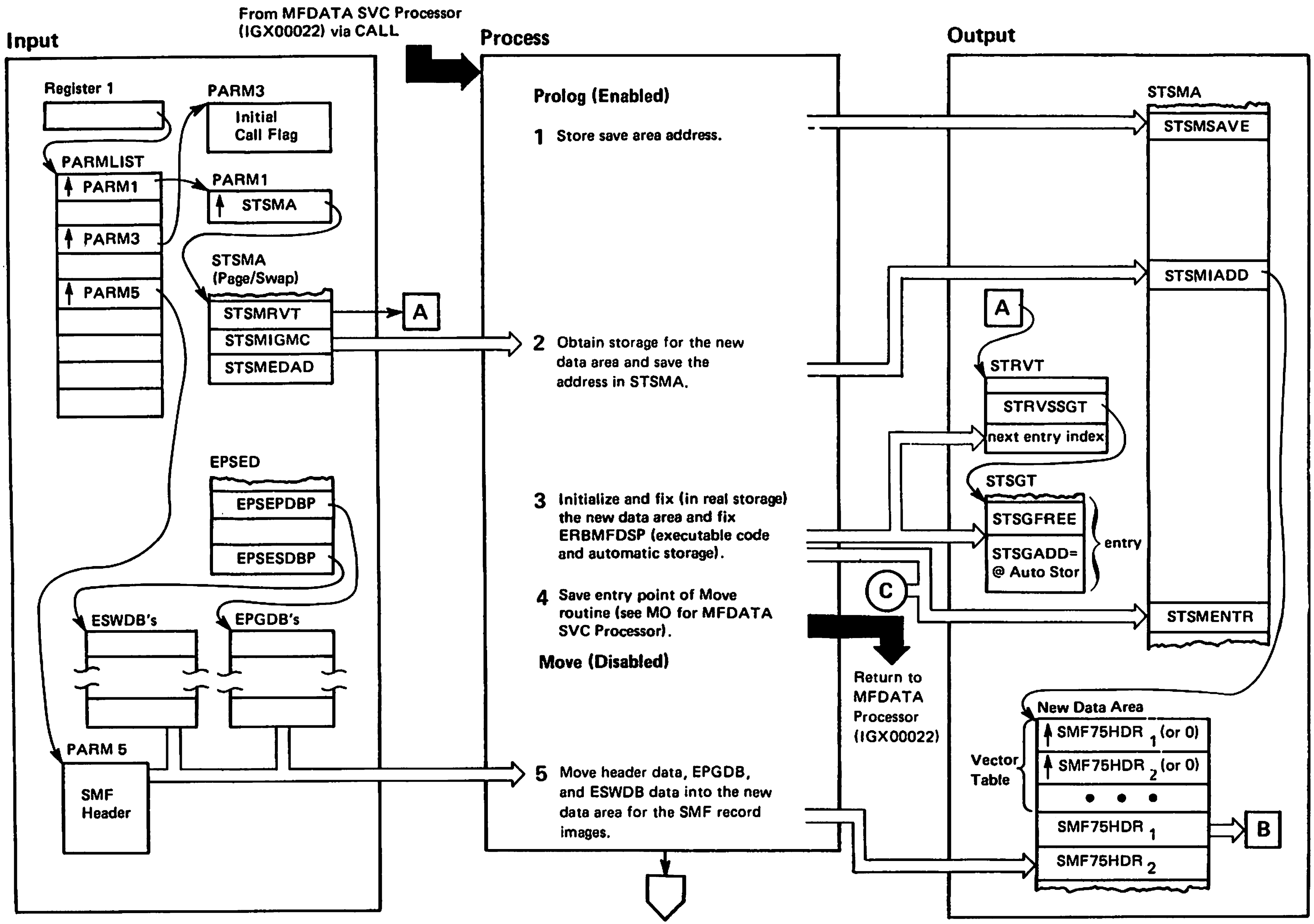


Diagram 41. Interval MG Routine for Trace (ERBMFDTP) (Part 6 of 6)

Extended Description	Module	Label
11 ERBMFDTP uses the PGSER macro to free ERBMFDTP and its automatic data, and saves the address of the SMF record pointer array in DTRETDTA.	ERBMFDTP	

Diagram 42. Interval MG Routine for Page/Swap Dataset Activity (ERBMFDSP) (Part 1 of 6)

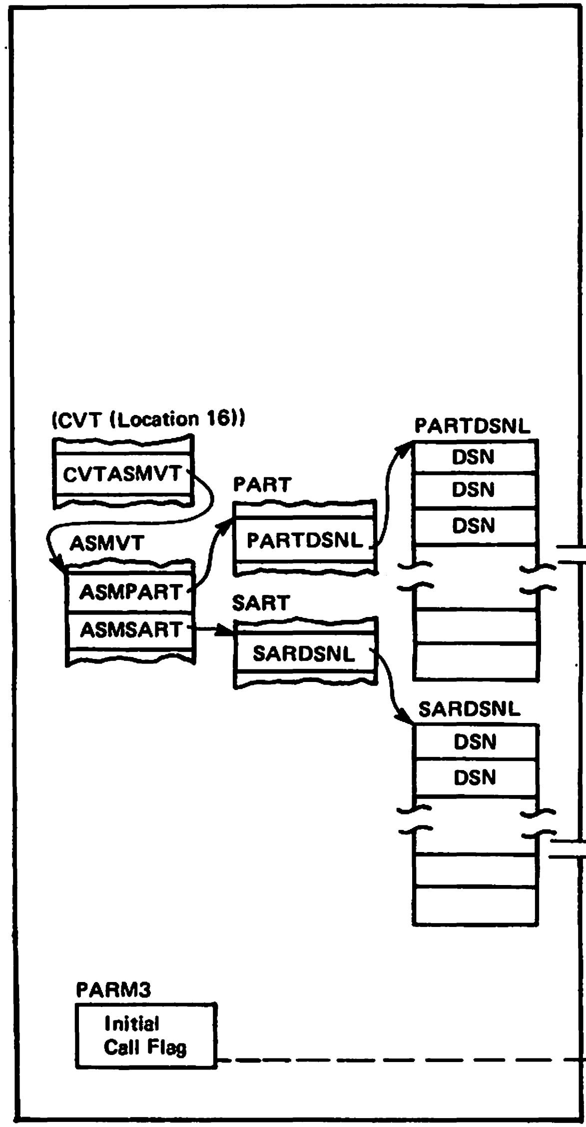


Extended Description	Module	Label
The interval MG routine for page/swap data set activity (ERBMFDSP) builds internal images of SMF-75 paging space records. If the RECORD option was specified when RMF was started, it writes images to the SMF data set. For these images, ERBMFDSP uses data contained in the page and swap event data table (EPSEDT) which was collected by ERBMFESP from the auxiliary storage manager (ASM). As described in the M.O. for the MFDATA SVC Processor, ERBMFDSP executes in three parts: PROLOG, MOVE, and EPILOG. No break in execution is apparent except for the need to save entry points for the MOVE and EPILOG parts.		
PROLOG (Enabled)		
1 On entry, the address of the standard save area is stored into the STSMSAVE field of the supervisor measurement area (STSMA). The address is later passed as a parameter on subsequent calls by IGX00022 when it reenters the MOVE and EPILOG sections.	ERBMFDSP	
2 ERBMFDSP uses the GETMAIN macro instruction to obtain storage in key zero. The request is made for storage in SP0 but because the request is made in supervisor state, the storage is actually obtained from SP252. This storage will contain the SMF Record images with the data for this interval. The address of this storage is saved in the STSMIADD field of the STSMA.	ERBMFDSP	
3 ERBMFDSP uses the PGSER macro instruction to inhibit paging of the data area, the routine ERBMFDSP, and its automatic storage. An entry is added to the storage resource table (STSGT) for the automatic storage, then the resource vector table (STRVT) is updated to indicate the index of the next available STSGT entry. **Note:** Pages are fixed to prohibit page fault interrupts during the disabled MOVE section.	ERBMFDSP	

Extended Description	Module	Label
4 The entry point is to be used by IGX00022 to enter the MOVE part of ERBMFDSP. Between the PROLOG and MOVE sections, a return is made to the caller that avoids freeing data that would be freed in normal returns.	ERBMFDSP	
MOVE (Disabled)		
5 ERBMFDSP moves an SMF record header and RMF product section into each interval record image (1 SMF record type 75 for each page data set and each swap data set). (ERBMFDSP fills in the slot usage for page data sets and swap set usage for swap data sets from fields kept in the page data set event data block (EPGDB) and the swap data set event data block (ESWDB). This usage data is as follows: • Slots/swap sets allocated • Maximum slots/swap sets used • Minimum slots/swap sets used • Number of bad slots/swap sets • Number of samples when the data set is in-use by ASM (auxiliary storage manager) • Number of requests against the data set • Number of samples ERBMFDSP calculates the average slots/swap sets used from the cumulative sum divided by the number of samples. ERBMFDSP saves the count of I/O requests in the SMF record for both the page and swap data sets and the count of pages transferred for the page data sets at the beginning of the interval. For local page data sets, it also saves an indicator showing whether or not the data set accepts VIO pages.	ERBMFDSP	

Diagram 42. Interval MG Routine for Page/Swap Dataset Activity (ERBMFDSP) (Part 3 of 6)

Input



Process

6 Reinitialize the Page and Swap Data Blocks (EPGDB and ESWDB).

7 Save the entry point of the Epilog routine. → C

Return to MFDATA Processor (IGX00022)

Epilog (Enabled)

8 Free the pages of previously fixed areas: new data area, ERBMFDSP, and its automatic storage.

9 Fill page data set and swap data set names into SMF Records.

10 If this is the first time through, return to the MFDATA processor. Otherwise, proceed to the next step.

Return (IGX00022)

Extended Description	Module	Label

The I/O Queuing Report Generator (ERBMFRQR) is passed interval data in the form of an internal SMF record image. It formats this data for its report and writes the report to a SYSOUT data set for either real-time or deferred printing (selected when RMF started).

1 Establish addressability for SMF record 78 including both its configuration section and data section.

2 The I/O Queuing Report Generator calls ERBMFRGMs internal procedure MFHDRISR to write the title 'I/O QUEUING ACTIVITY' on each new page.

3 The column headings are put into the internal page via ERBMFRGM's internal procedure MFISRTXT. The column headings are:

LCU
ACTIVITY RATE
AVG Q LNGTH
% All CH PATH BUSY
% REQ DEFER
– % REQ DEFER – DEV BUSY
– % REQ DEFER – CU BUSY
CONTROL UNITS
CHAN PATHS

Note that the data fields for AVG Q LNGTH are blank when RMF is running on a 4381 processor.

Extended Description	Module	Label

4 The following data is taken from the SMF record:

a. Logical control unit (LCU) number
b. Channel paths (CPIDs)
c. Physical control units (PCUs)

The following data is calculated from values in the SMF record:

d. Activity rate for successful initial selections (Activity Rate=R781SIS/SMF78INT)

e. Average queue length of requests on logical control unit queue (only for 308x processors) (Avg Q Length=R781QUE/R781TIS)

Because the I/O (CUCW) queue length is not available on 4381 processors, the R781QUE field is meaningless for 4381 processors, and the average queue length is not calculated.

f. Percentage of time when all channel paths belonging to an LCU were busy at the same time (% ALL CH PATHS BUSY=100*(R781ABY/SMF78SAM)

g. Percentage of unsuccessful selections (% REQ DEFER=100*((R781TIS-R78SIS)/R781TIS)

h. Percentage of unsuccessful selections due to device busy (% REQ DEFER DEV BUSY=100* (R78DVB/R781TIS))

i. Percentage of unsuccessful selections due to control unit busy (% REQ DEFER CU BUSY=100* (R781CUB/R781TIS))

Diagram 66. I/O Queuing Report Generator for 308x/4381 Processors (ERBMFRQR) (Part 3 of 4)

Input

Process

5 Convert data from binary to character strings for the report line.

ERBMFCNV

6 Insert the line into internal report page.

MFISRTXT

7 Determine if a page is full.

8 Return to caller.

MFWRTPAG

To Report Generator Control (ERBMFRGM)

Output

Print Page

Extended Description	Module	Label

5 ERBMFRQR calls module ERBMFCNV to convert the data gathered and calculated in the previous step from binary to the equivalent character strings. The resulting string is returned as a fixed length field.

The following are provided as input parameters to ERBMFCNV:

- The input binary value.
- The signed decimal scaling factor for the input value.
- The address of the output string.
- The number of the output string.
- The number of digits to the right of the decimal point.
- Commas or no commas.
- Floating point or no floating point.

If the data field is large enough to contain the converted value, the value is right justified.

If the data field is too small to contain the converted value, the following actions are taken (in the following order) to attempt to preserve valuable data.

- Editing commas are removed.
- The least significant digits to the right of the decimal point are removed, up to and including the decimal point.
- The field is replaced with asterisks.

The return code indicates which action was taken.

RC0 The value fits in the field, either with or without commas.

RC4 The least significant digits have been truncated.

RC8 Asterisks are being returned.

Extended Description	Module	Label

6 The insert text subroutine MFISRTXT moves the converted data into the page image. Depending upon the channel path status flag (R781CPST) containing the status change and online indicators (R781CPV, R781CPO), additional lines might be printed to provide the channel path identifier followed by the text, either "NOW ONLINE", or "NOW OFFLINE" or "OFFLINE".

Additional lines might also be printed, depending on the connectivity status flags (R781VP, indicating connectivity status changed, and R781VPOF, indicating path offline to all devices). Once a channel path is online to the system, the contents of the status flags can cause RMF to print the channel path identifier(s) followed by the text, "PATH(S) NOW ONLINE", "PATH(S) NOW OFFLINE", or "PATH(S) OFFLINE".

7 If there is still room in the internal page image (current line count less maximum line count) go to Step 4 to complete the page. If the page is full, call the MFWRTPAG subroutine to write out a page of the report.

MFWRTPAG writes the internal page image, line by line, to the SYSOUT data set using a QSAM PUT. Blank lines are consolidated and a single record is written with print control characters, indicating the number of lines to skip.

8 If ERBMFRQR finds more logical control units to report on (using the SMF78ASN count), control returns to Step 2 to repeat the processing for the next page of I/O queuing data. Otherwise, ERBMFRQR prints out the last page of the report, if any, and returns to the caller.

ERBMFRGR - MODULE DESCRIPTION

DESCRIPTIVE NAME: I/O Queuing Report Generator for IBM 3090 Processors

FUNCTION:

ERBMFRGR receives data in the form of an internal SMF record image. It then formats and prints the I/O queuing activity report for processors.

ENTRY POINT: ERBMFRGR

PURPOSE: See function

LINKAGE: BALR - from ERBMFRGM

CALLERS: ERBMFRGM - Report generator control

INPUT:
- Parameter 1 - Pointer to the data to be formatted and printed.
- Parameter 2 - Pointer to the subroutine vector table that contains the addresses of the following subroutines:
  - ERBMFCNV - binary to EBCDIC conversion
  - MFHDRISR - header insertion
  - MFISRTXT - insert text
  - MFWRTPAG - write internal page image
- Parameter 3 - Pointer to problem state measurement area for this report (STPMA).
- Parameter 4 - Entry code:
  - X'20' Post processor call

OUTPUT: I/O queuing activity report.

EXIT NORMAL: Return to caller.

EXTERNAL REFERENCES: See below.

ROUTINES:
- ERBMFRGM - (entry point MFHDRISR) header insertion
- ERBMFRGM - (entry point MFISRTXT) insert text into page image
- ERBMFRGM - (entry point MFWRTPAG) write internal page image

DATA AREAS: ERBMFLQV - I/O Queuing report language parts

CONTROL BLOCKS:
- ERBSMF78 - I/O queuing SMF record
- ERBMFLNG - Language parts table
- ERBMFLQM - Language parts constants
- ERBRGCON - Line and column maximum constants

ERBMFRGR - MODULE OPERATION

1. Calculates and saves the interval time in milliseconds.

2. Loops through the IOP (input/output processor) initative queue data sections. Computes and saves the IOP report data for each header line.

3. Issues the report header lines and the IOP data.

4. Depending on global information, issues text 'DIAGNOSE INTERFACE FAILURE', 'CHANNEL MEASUREMENT FACILITY NOT ACTIVE', or 'NO ACTIVITY FOR SELECTED LCUs'.

5. Depending on channel path status, inserts status text.

6. Computes and inserts a line of data for each channel path belonging to a selected LCU.

7. Depending on device path connectivity, inserts status text.

8. If a page is filled or all LCUs have been processed, writes the current page of the report.

9. When all selected LCUs have been reported, returns to the caller.

RECOVERY OPERATION:

Error recovery for this module is accomplished externally by module ERBMFSAR.

ERBMFRGR - DIAGNOSTIC AIDS

ENTRY POINT NAME: ERBMFRGR

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES: None

REGISTER CONTENTS ON ENTRY:

```
Register  0    - irrelevant
Register  1    - parameter list address
Register 2 -12 - irrelevant
Register 13    - save area address
Register 14    - return address
Register 15    - entry point address
```

REGISTER CONTENTS ON EXIT: Irrelevant

ERBMFRGM - Report generator control

PARAMETERS

ERBMFRGR

RGDATAVP RGVTBLVP
RGPMAVP RGECDEVB

RGPPCLCB

PARAMETERS

RGECDEVB

RG0001CF RGONCB
RGCINTVF

ERBSMF78

SMF78INT

ERBMFRGR receives data in the form of an internal SMF record image. It then formats and prints the I/O queuing activity report for processors.

01 Initializes processing control values snd calculates interval time in milliseconds. (If ERBMFRGR has been invoked during post processor execution, the post processor has already calculated the interval time.)

RGPCNTVF
RGHINSVB
RGCINTVF

02 Processes IOP values.

If the DIAGNOSE interface failure bit (R783GDIF) is off and the channel measurement facility bit (R783GCMF) is off, ERBMFRGR processes the IOP data for each report header line.

ERBSMF78

R783GD R783IQI

SVGDIF RGOFFCB
SVGCMF RGIOPXVF

ERBSMF78

R783GFLG R783GIDS
R783GIDN R783GSAM
R783IQID

SVIQDN RGONCB
RGIOPVC RGTRTCC
RGWSTRVC

A. Loops through all IOP initiative queue data sections. If IOP is installed (R783IQI = ON), ERBMFRGR saves the IOP values for later use.

SVGFLAG
SVIQDN
SVGSAM
RGIOPPVP
RGIOPXVF
SVIOPIQI
RGIOPVC
RGWSTRVC
SVIOP

RGCINTVF RG0000CF

B. Calculates and saves the activity rate. The formula is:
R783IQCT
-------- = ACTIVITY RATE
RGCINTVF
If the interval length (RGCINTVF) is zero, inserts an asterisk (*) as the activity rate value.

C. Converts the activity rate value

ERBMFCNV

RGWVALVF, RGMIN3CF,
RGWVALVC, RG0008CF,
RG0003CF, RGNOCMCB,
RGNOFLCB, RGCWRKVC

RGIOPXVF

RGWVALVC RGASTXCC

SVIOPAR

RG0000CF

ERBSMF78

R783IQSM R783IQCT

D. Calculates and saves the average queue length.
The formula is:
R783IQSM - R783IQCT
------------------ = AVG Q LNGTH
R783IQCT
If the total number of requests enqueued (R783IQSM) is zero, inserts an asterisk (*) as the average queue length value.

RGAVALVF

E. Converts the average queue length value.

ERBMFCNV

RGWVALVF, RGMIN6CF,
RGWVALVC, RG0005CF,
RG0002CF, RGNOCMCB,
RGNOFLCB, RGCWRKVC

RGIOPXVF

ERBSMF78

R783IQD R783IQSM
R783IQCT

RGWVALVC RG0000CF
RGASTXCC RGOFFCB
RGIOPPVP

SVIOPAQL
RGWVALVF
SVIOPIQI
RGIOPPVP

ERBSMF78

SMF78ASN

RG0000CF

03 Inserts the report header lines.

A. Calls the header insertion subroutine.

RGHDRIN: 30

04 If the number of LCUs (SMF78ASN) is zero, provides one of the following texts instead of a report.

SVGDIF RGONCB

A. If the DIAGNOSE interface failed, inserts 'DIAGNOSE INTERFACE FAILURE'.

MFISRTXT

LNGTPTR(LQPART40)->LNGTXTD,
LNGTPTR(LQPART40)->LNGTXTL,
RGLCNTVF, RGTOTSCF

SVGCMF RGONCB

B. If the channel measurement facility is not active inserts 'CHANNEL MEASUREMENT FACILITY NOT ACTIVE'.

MFISRTXT

LNGTPTR(LQPART38)->LNGTXTD,
LNGTPTR(LQPART38)->LNGTXTL,
RGLCNTVF, RGTOTSCF

C. If the number of selected LCUs is zero, inserts 'NO ACTIVITY FOR SELECTED LCUs'.

MFISRTXT

LNGTPTR(LQPART37)->LNGTXTD,
LNGTPTR(LQPART37)->LNGTXTL,
RGLCNTVF, RGTOTSCF

ERBSMF78

R783CS R783DS

ERBSMF78

SMF78ASN

05 If the number of LCUs is not zero, processes each LCU, reported by LCU number in ascending order. ERBMFRGR loops through all configuration and data sections.

RGLCUPVP
RGLCUDVP
RGLCUXVF

RGHINSVB RGONCB

A. If a new header is required, calls the header insertion subroutine.

RGHDRIN: 30

ERBSMF78

R783ID1

RGLCUNVC RGTRTCC

RGLCUNVC
RGWSTRVC

B. Inserts the LCU number

MFISRTXT

RGWSTRVC, RG0003CF,
RGLCNTVF, RGLCUCF

ERBSMF78

R783NHMA

RGOFFCB

06 Uses model-dependent data to calculate the contention rate and delay queue length for the LCU. If the model dependent data items for the LCU are invalid (R783NHMA = ON), skips the calculations.

RGCINTVF RG0000CF

A. Calculates and inserts the contention rate.
The formula is:

```
R783QCT
-------- = CONTENTION RATE
RGCINTVF
```

B. Converts the contention rate value.

ERBMFCNV

RGWVALVF, RGMIN3CF,
RGWVALVC, RG0008CF,
RG0003CF, RGNOCMCB,
RGNOFLCB, RGCWRKVC

C. Inserts the contention rate value.

MFISRTXT

RGWVALVC, RG0008CF,
RGLCNTVF, RGCCF+1

D. If the interval length (RGCINTVF) is zero, inserts an asterisk (*) as the contention rate value.

MFISRTXT

LNGTPTR(LQPART27)->LNGTXTD,
LNGTPTR(LQPART27)->LNGTXTL,
RGLCNTVF, RGCCF+1

RG0000CF

ERBSMF78

R783QSM R783QCT

E. Calculates and inserts the delay queue length.
The formula is:

```
R783QSM - R783QCT
----------------- = DELAY Q LNGTH
  R783QCT
```

RGAVALVF

F. Converts the delay queue length value.

ERBMFCNV

RGWVALVF, RGMIN6CF,
RGWVALVC, RG0005CF,
RG0002CF, RGNOCMCB,
RGNOFLCB, RGCWRKVC

G. Inserts the delay queue length value.

MFISRTXT

RGWVALVC, RG0005CF,
RGLCNTVF, RGDCF

ERBSMF78

R783QSM R783QCT

RG0000CF

RGWVALVF

H. If the number of queue samples (R783QCT) is zero inserts an asterisk (*) as the delay queue length value.

MFISRTXT

LNGTPTR(LQPART27)->LNGTXTD,
LNGTPTR(LQPART27)->LNGTXTL,
RGLCNTVF, RGDCF

I. If the model dependent data items for the LCU are invalid, inserts text 'NO H/W DATA'.

MFISRTXT

LNGTPTR(LQPART39)->LNGTXTD,
LNGTPTR(LQPART39)->LNGTXTL,
RGLCNTVF, RGCCF

RG0000CF RGPBAVVB
RGNMCDVB

ERBSMF78

R783CPD R783CPO
R783CPV R783VPOF
R783VP R783NMCD

GPR00F RG100TCF
RGOFFCB RGLCUPVP
RGONCB SVGSAM
RGCPDPVP GPR01F

ERBSMF78

R783CPDS R783CHPN
R783PB

07 Loops through all channel paths in this LCU to calculate and insert the percentage value for ALL CH PATH BUSY.
The formula is:
(path busy count (CHP1) / sample count) *
(path busy count (CHP2) / sample count) *
(path busy count (CHP3) / sample count) *
(path busy count (CHP4) / sample count) =
percent all channel path busy.

A. If the measured configuration data is not available (RGNMCDVB = ON) and no percent busy values are available, (RGPBAVVB = OFF), leaves the ALL CH PATH BUSY field blank.

B. Converts the percent all channel path busy value.

ERBMFCNV

RGWVALVF, RGMIN3CF,
RGWVALVC, RG0006CF,
RG0002CF, RGNOCMCB,
RGNOFLCB, RGCWRKVC

C. Inserts the percent all channel path busy value.

MFISRTXT

RGWVALVC, RG0006CF,
RGLCNTVF, RGACF-1

GPR00F
GPR01F
RGPBAVVB
RGNMCDVB
RGCPDPVP
RGCPDXVF
GPR15F
GPR03F
RGWVALVF

RGVONCB RGCPDXVF
RGONCB RGWSEPVF

ERBSMF78

R783CPI

ERBMFLQM

LQPART19

RGLCUPVP RGCPPFVB
RGCUWSVC RGCUTXVF
RGCUSPVF LNGTPTR
LNGTXTD RG0001CF
RGCUPVF RGCUVC
RGTRTCC RG0003CF
RGWSTRVC

ERBSMF78

R783CPDS R783CHPN
R783CUN R783CU

08 Inserts the control unit numbers and data for each channel path associated with this LCU.

A. Inserts the control unit number(s).

MFISRTXT
RGCUWSVC, RGWSEPVF, RGLCNTVF, RGCTRCF

RGCPDPVP
RGCPPFVB
RGCPDXVF
RGCUSVC
RGCUTXVF
RGCUWSVC
RGCUSPVF
RGCUPVF
RGCUVC
RGWSTRVC
RGWSEPVF

RGCHPXVF RGONCB
RGCUSVC

ERBSMF78

R783CPI R783CU

RGCPDPVP RGCPDXVF
RGCPPFVB RGVONCB

ERBSMF78

R783CHPN

B. Loops through the remaining CHPIDs to detect a CU string that matches the one previously processed for this LCU.

C. Calls subroutine RGCPIDIN to complete a report line for this channel path ID.

RGCPIDIN: 28

RGCHPPVP
RGCHPXVF
RGCPPFVB

ERBSMF78

R783CPD

RGCHPPVP RGCPDPVP

RGCHPPVP
RGCPDPVP

RGLCNTVF

ERBRGCON

RGMAXLN

09 If there is still room in the internal page image (current line count, less maximum line count - 8), repeats processing for the next line of I/O queuing data.
Otherwise ERBMFRGR calls the procedure MFWRTPAG in ERBMFRGM to write out a page of the report.

MFWRTPAG

RGONCB

RGLCNTVF RG0000CF

RGLCUPVP RGLCUDVP

ERBSMF78

SMF78DCL SMF78ASL

10 If ERBMFRGR (using the SMF78ASN count) finds more logical control units to report, repeats the processing for the next page of I/O queuing data. Otherwise ERBMFRGR prints out the last page of the report, if any, and returns to the caller.

11 Returns to the caller.

RGHINSVB

RGLCUPVP
RGLCUDVP

ERBSMF78

R783CPID

RGCPIDVC RGTRTCC

12

RGCPIDIN

12 Inserts the channel path data.
Places the current channel path ID in the work string and sets up loop control to search for other channel path ID's with the same set of control units.

A. Inserts channel paths.

MFISRTXT
RGWSTRVC, RG0002CF, RGLCNTVF, RGCHCF+1

RGCPIDVC
RGWSTRVC

RGONCB

ERBSMF78

R783CPI R783CPO
R783CPV

RGOFFCB

B. If a channel path has been configured online during this interval, inserts 'NOW ONLINE'.

MFISRTXT
LNGTPTR(LQPART20)->LNGTXTD, LNGTPTR(LQPART20)->LNGTXTL, RGLCNTVF, RGCUBF+7

RGNTXTVB
RGFTXTVB
RGTXTVB

RGOFFCB

ERBSMF78

R783CPI R783CPO
R783CPV

RGONCB

C. If a channel path has been configured offline during this interval, inserts 'NOW OFFLINE'.

MFISRTXT

LNGTPTR(LQPART21)->LNGTXTD,
LNGTPTR(LQPART21)->LNGTXTL,
RGLCNTVF, RGCUBF+7

RGNTXTVB

RGOFFCB

ERBSMF78

R783CPI R783CPO
R783CPV

RGONCB

D. If a channel path has been offline without configuration activity during this entire interval, inserts 'OFFLINE'.

MFISRTXT

LNGTPTR(LQPART22)->LNGTXTD,
LNGTPTR(LQPART22)->LNGTXTL,
RGLCNTVF, RGCUBF+7

RGFTXTVB

RGONCB

RGFTXTVB
RGTXTVB

13 **If the channel path was not offline during the entire interval, calculates and inserts the measured channel path data.**

ERBSMF78

R783VPOF R783VP
R783NMCD

RGONCB RGOFFCB
RGTXTVB RGCINTVF
RG0000CF

A. Calculates and inserts CHPID TAKEN. The formula is:

R783PT
-------- = CHPID TAKEN
RGCINTVF

B. Converts the channel path taken value.

ERBMFCNV

RGWVALVF, RGMIN3CF,
RGWVALVC, RG0006CF,
RG0003CF, RGNOCMCB,
RGNOFLCB, RGCWRKVC

C. Inserts the channel path taken value

MFISRTXT

RGWVALVC, RG0006CF,
RGLCNTVF, RGCPCF-1

D. If the interval length (RGCINTVF) is zero, inserts an asterisk (*) as the channel path taken value.

MFISRTXT

LNGTPTR(LQPART27)->LNGTXTD,
LNGTPTR(LQPART27)->LNGTXTL,
RGLCNTVF, RGCPCF-1

RG0000CF

ERBSMF78

R783CUB R783PT

E. Calculates and inserts the percent CU BUSY value.
The formula is:

```
 R783CUB
---------------- = percent CU BUSY
R783PT + R783CUB
```

RGAVALVF

F. Converts the percent control unit busy value.

ERBMFCNV

RGWVALVF, RGMIN4CF,
RGWVALVC, RG0006CF,
RG0002CF, RGNOCMCB,
RGNOFLCB, RGCWRKVC

G. Inserts the percent control unit busy value.

MFISRTXT

RGWVALVC, RG0006CF,
RGLCNTVF, RGCUCF-2

ERBSMF78

R783CUB R783PT

RG0000CF

RGWVALVF

H. If the count of channel path taken (R783PT) is zero, inserts an asterisk (*) as the percent control unit busy value.

MFISRTXT

LNGTPTR(LQPART27)->LNGTXTD,
LNGTPTR(LQPART27)->LNGTXTL,
RGLCNTVF, RGCUCF-2

RGFTXTVB RGOFFCB
RGONCB RGNTXTVB

ERBSMF78

R783VPOF R783VP

RGLCNTVF RG0001CF

RGLCNTVF

I. If the channel path was online to the system but had previously had connectivity to any device in the LCU, insert 'PATH NOW ONLINE'.

RGFTXTVB RGOFFCB
RGONCB RGNTXTVB

ERBSMF78

R783VPOF R783VP

RGLCNTVF RG0001CF

J. If the channel paths was online to the system but no longer has any connectivity to any device in this LCU, insert 'PATH NOW OFFLINE'.

RGLCNTVF

RGFTXTVB RGOFFCB
RGONCB RGNTXTVB

ERBSMF78

R783VPOF R783VP

RGLCNTVF RG0001CF

K. If a channel path is online to the system but has no connectivity to any device of this LCU, inserts 'PATH OFFLINE.'

RGLCNTVF

14
RGHDRIN

14 Builds new header.

A. Inserts the report header.

MFHDRISR

RGDATAVP,
LNGTPTR(LQPART01)->LNGTXTD,
LNGTPTR(LQPART01)->LNGTXTL,
RGPCNTVF DA
E hRGPCNTVF

RGPCNTVF RG0001CF
RGLN09CF

RGLCNTVF

B. Converts the value for the total number of samples.

ERBMFCNV

SMF78SAM, RG0000CF,
RGWVALVC, RG0006CF,
RG0000CF, RGNOCMCB,
RGNOFLCB, RGCWRKVC

C. Inserts the value for the total number of samples.

MFISRTXT

RGWVALVC, RG0006CF,
RGLN07CF, RGSAMPCF

Input: SVGDIF, RGOFFCB, SVGCMF, SVIOPIQI, RGIOPXVF, RGONCB

Input: SVIQDN

Output: RGIOPXVF

D. Inserts the IOP id.

MFISRTXT
SVIOPID(RGIOPXVF), RG0002CF, RGLCNTVF, RGIOPCF+1

E. Inserts the value for the IOP activity rate.

MFISRTXT
SVIOPAR(RGIOPXVF), RG0008CF, RGLCNTVF, RGIACF+2

F. Inserts the value for the IOP average queue length.

MFISRTXT
SVIOPAQL(RGIOPXVF), RG0005CF, RGLCNTVF, RGIAQCF+2

Input: RGLCNTVF, RG0001CF

Output: RGLCNTVF

Input: RGHDRTCF

Output: RGCLMXVF

G. Builds the header for the report columns.

MFISRTXT
LNGTPTR(RGPARTCF(RGCLMXVF))->LNGTXTD, LNGTPTR(RGPARTCF(RGCLMXVF))->LNGTXTL, RGHDRLCF(RGCLMXVF), RGHDRCCF(RGCLMXVF)

Input: RGOFFCB, RGLN17CF

H. Turns off the header insertion indicator.

Output: RGHINSVB, RGLCNTVF

15 Returns to the subroutine caller.

Diagram 68. Virtual Storage Report Generator (ERBMFRVR) (Part 1 of 2)

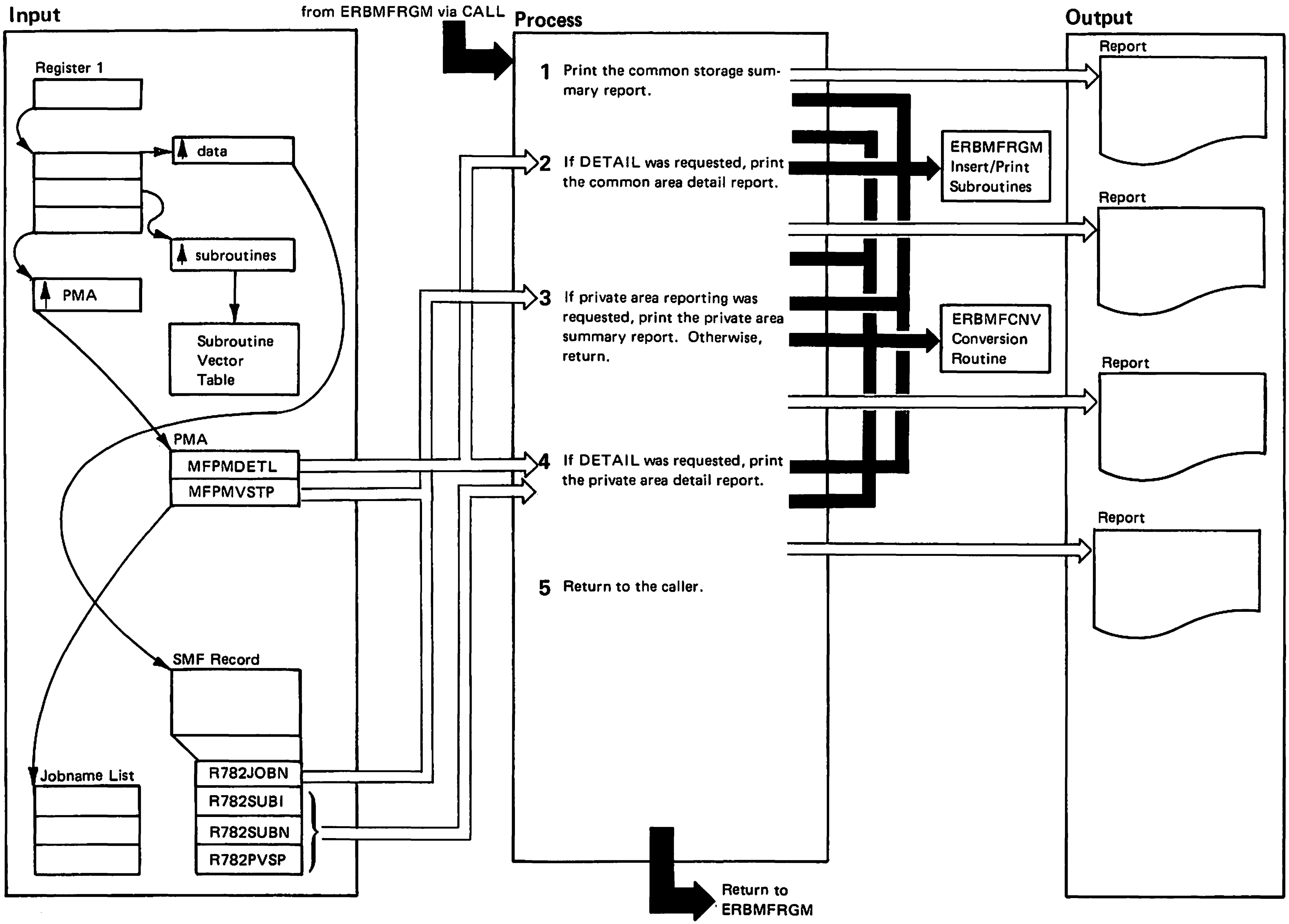


Diagram 68. Virtual Storage Report Generator (ERBMFRVR) (Part 2 of 2)

Extended Description	Module	Label
ERBMFRVR, which is called by ERBMFRGM, formats and prints the virtual storage report. It reports the data collected in the type 78 subtype 2 SMF record. It computes average values by using a running total of values from each sample and dividing this total by the number of samples. This calculation is a floating point calculation.	ERBMFRVR	ERBMFRVR

1 Formats the static storage section, the allocated CSA/SQA section, the wasted space section, the CSA/SQA free space section, and the maximum user region section, using data from R78COMN.

2 If a detail report is requested (the MFPMDETL bit is on), formats, by subpool section, the common storage detail section.

3 Scans each private area section (R782PVSP) indicated in SMF78ASN. If the report request specified no job selection list or if the job is in the list, formats the private area summary section, the private area storage map, and the free space section.

4 For each job selected in Step 3, checks the MFPMDETL bit; if the bit is on, formats the private area subpool section. R782SUBI and R782SUBN index the first and last subpool sections (R782PVSP) for this job.

5 Return to the caller.

This module has no special recovery processing. ERBMFRGM handles any errors that occur.

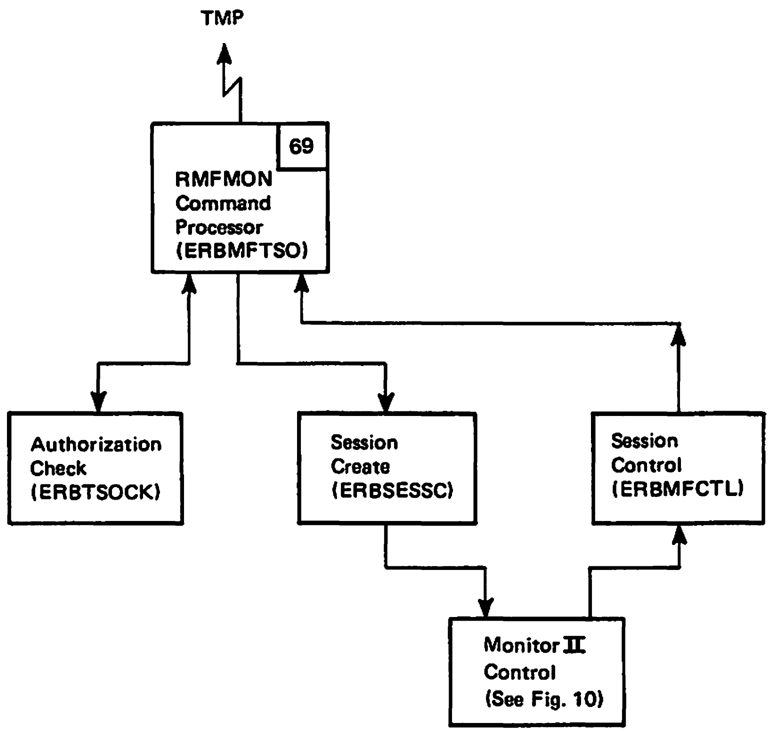


Figure 9. Monitor II TSO Session Processing Overview

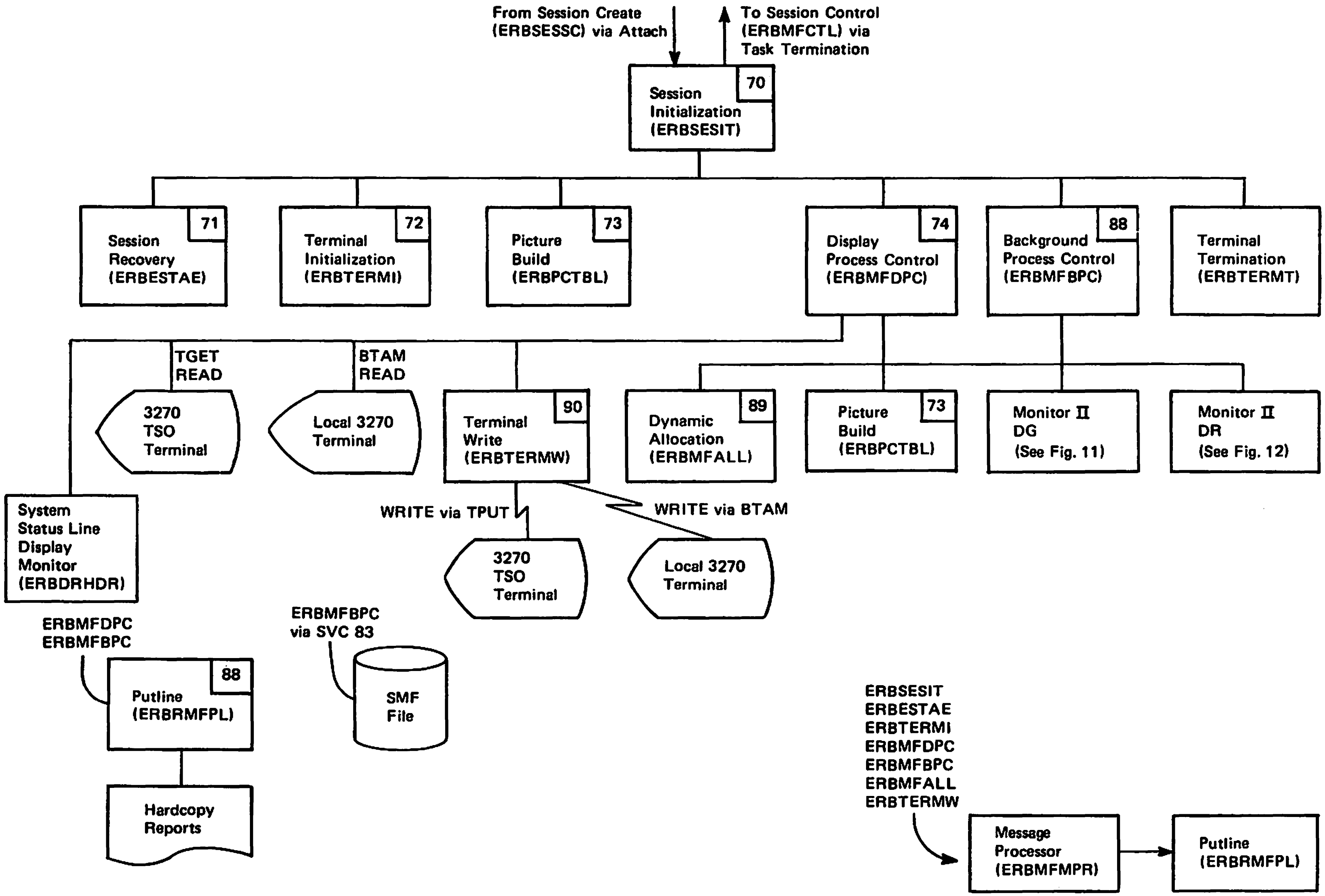


Figure 10. Monitor II Control Overview

Diagram 69. RMFMON Command Processor (ERBMFTSO) (Part 1 of 2)

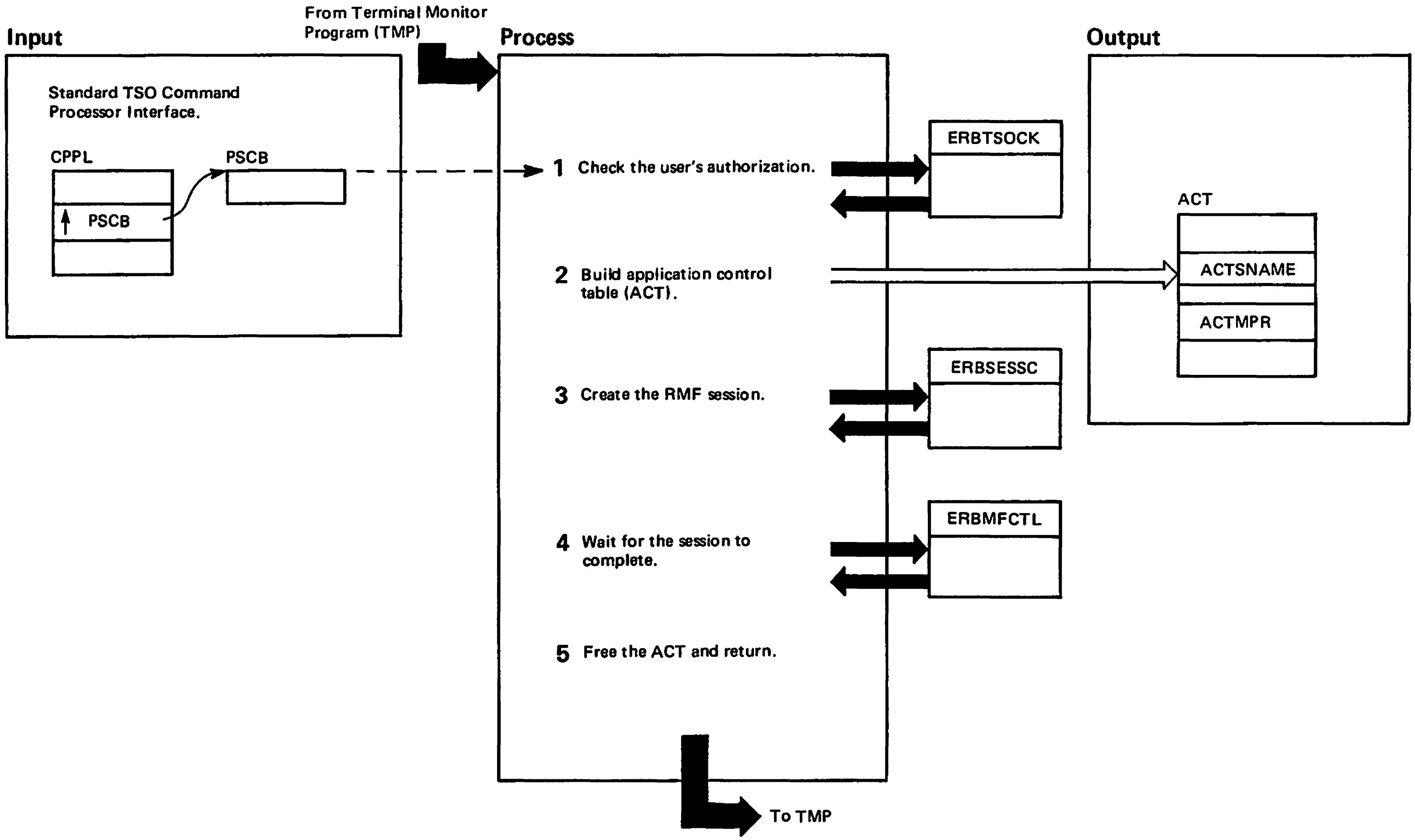
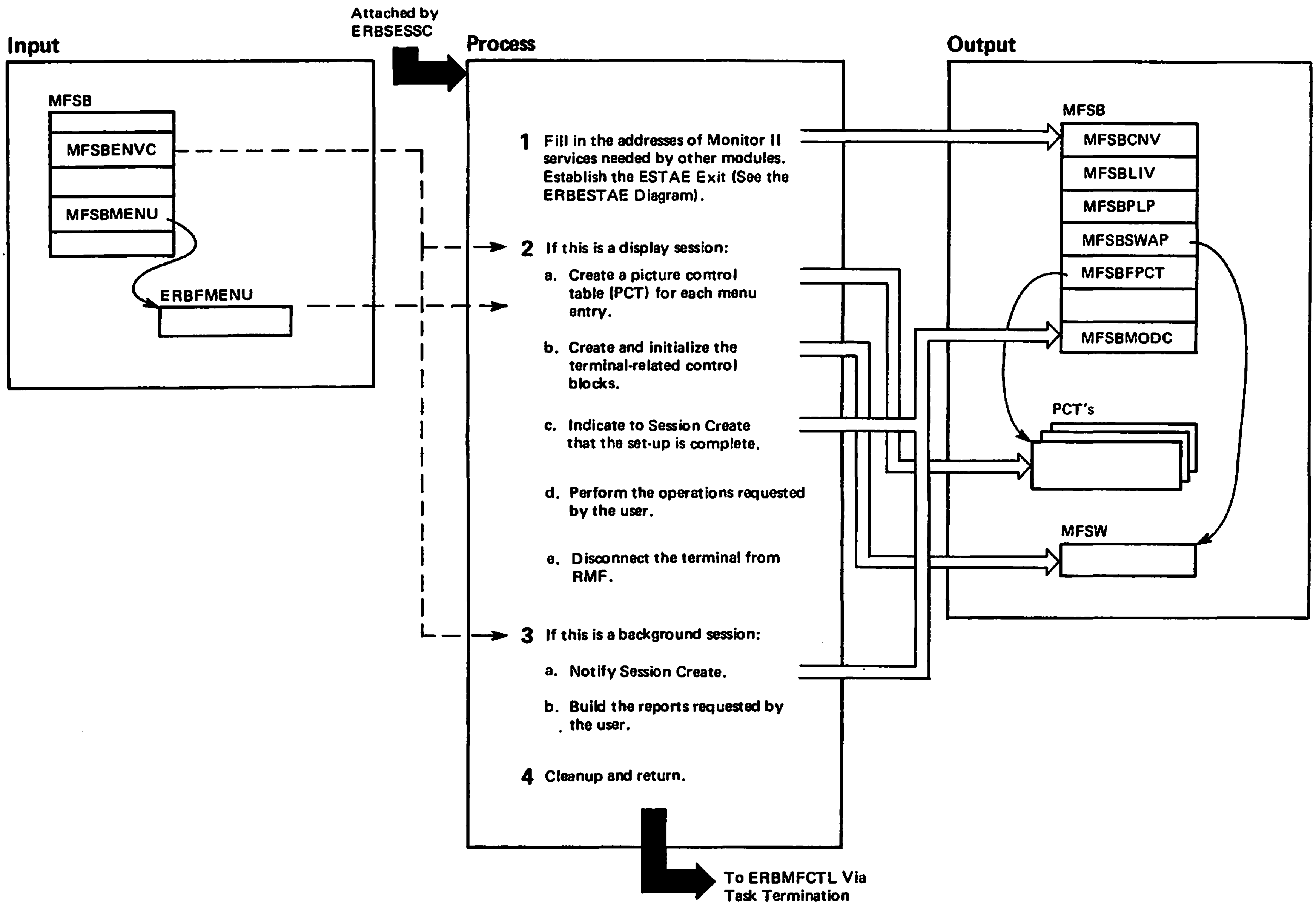


Diagram 69. RMFMON Command Processor (ERBMFTSO) (Part 2 of 2)

Extended Description	Module	Label
The purpose of ERBMFTSO is to provide enough of an RMF-like environment to allow a display monitor session to run. This consists mainly of creating an ACT (Application Control Table) and generating a session name of 'TSO'. It then calls routines that create and control the session. These routines are the same ones that perform these functions for the RMF started task.		ERBMFTSO
1 Check the user's authorization by calling an installation-replaceable module ERBTSOCK. Pass as parameters the userid and the PSCB (Protected Step Control Block), a TSO control block. The PSCB contains a copy of a field in the UADS (TSO User Attribute Dataset) defined for installation use. An installation could use this field for authorizing RMFMON. If authorization is denied (indicated by return code > 0 from ERBTSOCK), a message is issued and the command processor terminates.		
2 The ACT is the top of the RMF control block structure. Enough of it is filled in to allow a TSO RMF session to operate. The session name (ACTSNAME) is 'TSO'. The address of the message processor (ERBMFMPR) is stored in ACTMPR.		

Extended Description	Module	Label
3 Session create (ERBSESSC) is called to create the RMF display session, that is, to attach ERBSESIT and wait until it initializes itself.	ERBSESSC	
4 Call session control (ERBMFCTL) to wait for the completion of the daughter task. This corresponds to the control of a display session in the RMF memory.	ERBMFCTL	
5 Control is returned from ERBMFCTL when the user entered 'Z' to terminate the session or when the daughter task abended and did not recover.		

Diagram 70. Monitor II Session Initialization (ERBSESIT) (Part 1 of 2)



1 Fill in the addresses of Monitor II services needed by other modules. Establish the ESTAE Exit (See the ERBESTAE Diagram).

2 If this is a display session:

a. Create a picture control table (PCT) for each menu entry.

b. Create and initialize the terminal-related control blocks.

c. Indicate to Session Create that the set-up is complete.

d. Perform the operations requested by the user.

e. Disconnect the terminal from RMF.

3 If this is a background session:

a. Notify Session Create.

b. Build the reports requested by the user.

4 Cleanup and return.

Diagram 70. Monitor II Session Initialization (ERBSESIT) (Part 2 of 2)

Extended Description	Module	Label
ERBSESIT is attached by Session Create and completes the Monitor II session initialization at the daughter task. It invokes the proper process controller and then cleans up prior to task termination.		
1 The connect routine (ERBMFCNV in MFSBCNV) is used by IBM-supplied reporters. The language part table (ERBMFLIV in MFSBLIV) is used by Display Process Control. Putline (ERBMFPL in MFSBPLP) is used by everyone.	ERBSESIT	
2 This is a displa, session if MFSBFDM or MFSBTSO in MFSBENVC is on.	ERBSESIT	
a. ERBPCTBL is called for each menu entry to create a picture control table (PCT), the internal representation of the measurement. If no PCT's are created, then message ERB203I is produced and the session is terminated.		
b. ERBTERMI is called to create and initialize the terminal related control block, screen workarea (MFSW).		
c. If no errors, then MFSBMODC, the modify complete ECB, is posted to allow session create (ERBSESSC) to handle other sessions.		
d. ERBMFDPC is called to communicate with the user and perform the operation requested. Control is returned when the user or the operator requests the session be terminated.		
e. ERBTERMT is called to undo what ERBTERMI has done. The data gatherers and reporters that were used this session have their addresses stored in the PCT. These addresses are 0, if they were not loaded.		

Extended Description	Module	Label
3 This a background session if MFSBBDM in MFSBENVC is on.	ERBSESIT	
a. Post MFSBMODC, the modify complete ECB, so session create (ERBSESSC) can proceed and can handle other sessions.		
b. ERBMFBPC, background process control, is called to control the execution of the pictures requested by the input data. It will return when the work requested is completed or when the operator stops the session.		
4 Cleanup the resources obtained during the session. MFSBCLUP is a word of flags indicating what has been obtained. • Delete ERBMFCNV. • If MFSBHOPN in MFSBCLUP is on, then close the hardcopy DCB. • If MFSBHALL in MFSBCLUP is on, then unallocate the hardcopy dataset. • Cancel the ESTAE. Return, which terminates the task.	ERBSESIT	

Diagram 71. Monitor II Recovery (ERBESTAE) (Part 1 of 2)

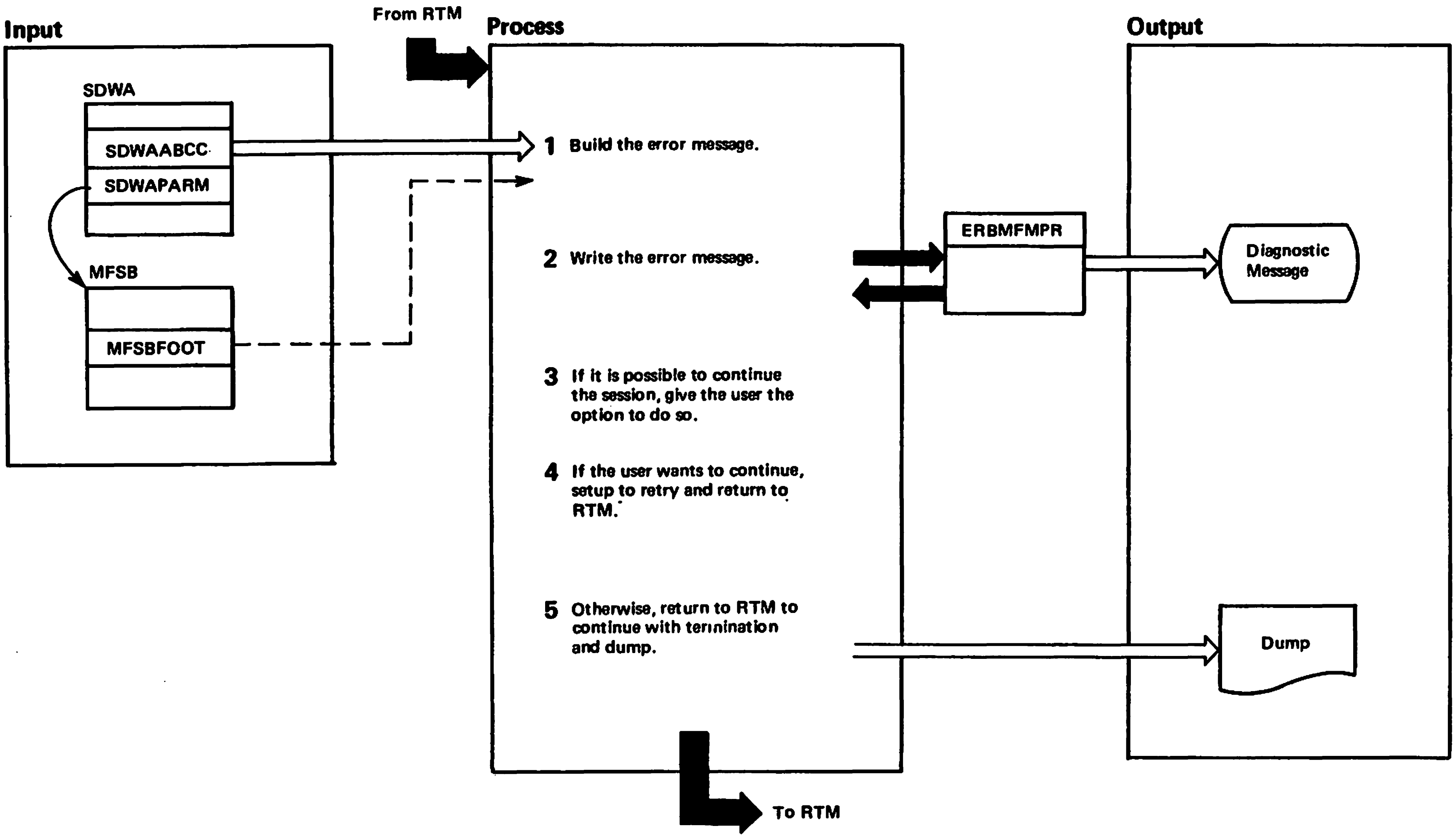


Diagram 71. Monitor II Recovery (ERBESTAE) (Part 2 of 2)

Extended Description	Module	Label
This ESTAE exit provides recovery for all Monitor II sessions.		
1 If this is a user abend, select a message from a table according to the completion code. If this is a system abend, build a message which includes the completion code and the module in control (as determined by footprints in MFSBFOOT.)		
2 Call ERBMFMPR, the message processor, to put out the message.	ERBMFMPR	
3 It is possible to continue the session if: • Display session initialization is complete (MFSBDPC is on), • No one has requested termination (MFSBTF and MFSBETF are off), and • SDWA was obtained. If it is possible to continue, give the user two options. 1. Continue the session. 2. End the session and get a dump.		
4 If the user wants to continue, reset all bits in MFSBFOOT e··cept ERBSESIT's and ERBESTAE's. Tell RTM to retry just after the read in ERBMFDPC's main loop. (This module has already read the user's next command.) Return to RTM.		
5 Otherwise, return to RTM to continue with termination and dump.		

Diagram 72. Terminal Initialization (ERBTERMI) (Part 1 of 2)

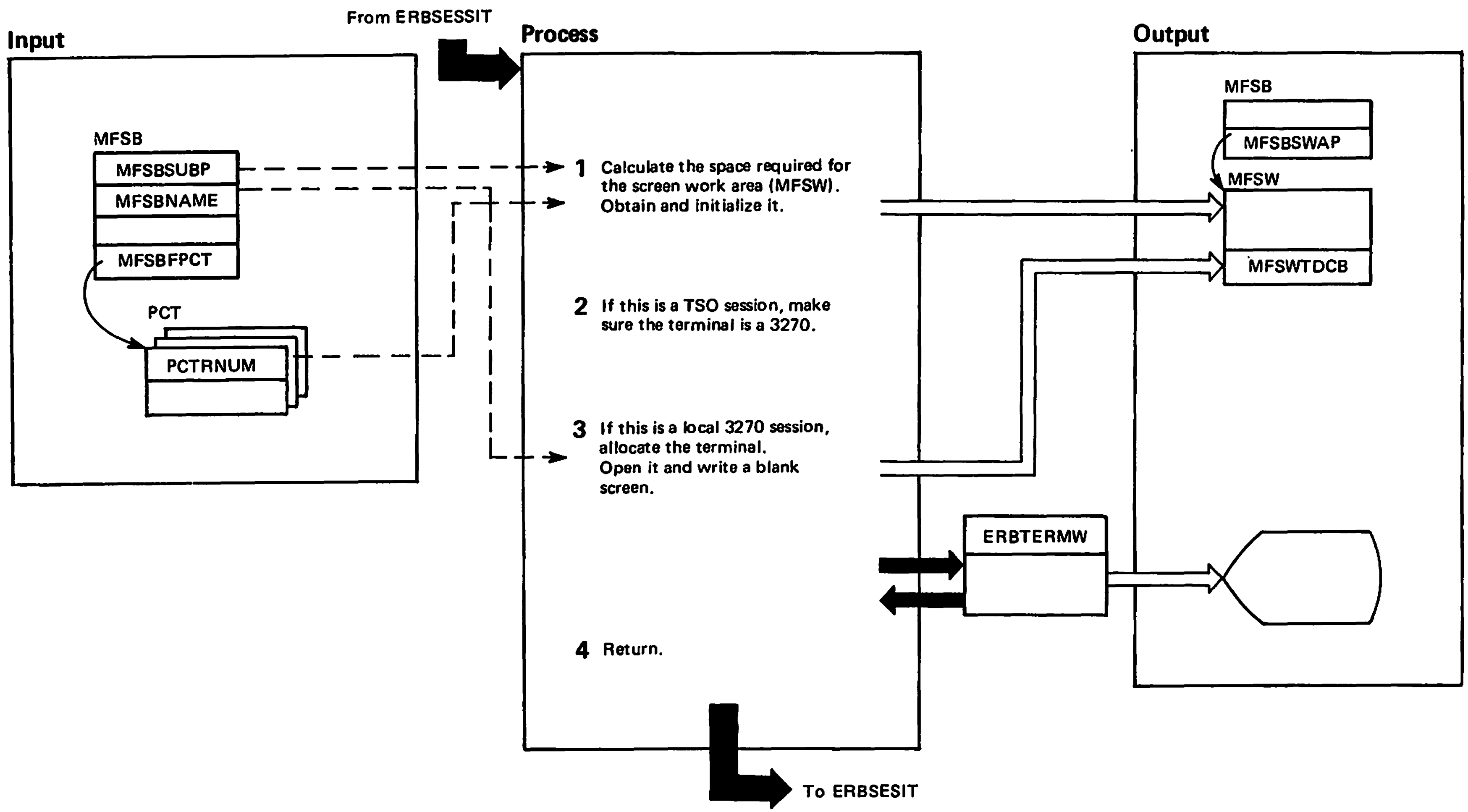


Diagram 72. Terminal Initialization (ERBTERMI) (Part 2 of 2)

Extended Description	Module	Label
1 The screen work area (MFSW) must have space for a logical buffer large enough to contain the maximum expected output. This is determined by using the largest PCTRNUM value on the PCT chain (number of relocate blocks to be generated by the picture). Make sure the largest PCTRNUM value is at least as large as the number of data lines on the screen. Obtain the storage from the session subpool (MFSBSUBP). The MFSW also has space for the output buffer, referred to as the physical buffer.	ERBTERMI	

The MFSW contains all the data required to communicate with the terminal, the DCB (for BTAM) and the input and output buffers. The output buffer, referred to as the physical buffer, is the one to which TPUTs (or WRITEs) are directed. The MFSW also contains a logical output buffer, which is larger and will usually contain more data than the physical buffer. Data is placed in the logical buffer by the display monitor commands that call ERBPUTSM, and by data reporters that call ERBRMFPL (which then calls ERBPUTSM). After data has been collected in the logical buffer, the display monitor calls ERBTERMW to move it to the physical buffer and TPUT or WRITE it. For the input, message, status and header areas, all the data is moved from the logical to the physical buffer.

However, the data area in the physical buffer has room for only 21 lines and the logical buffer may contain more. ERBTERMW begins moving with the line indicated by MFSWDFLI, current data frame index, and moves as many lines as will fit. This value starts at 1, is increased by the frame command, and reset to 1 by ERBRESET.

ERBTERMI initializes all the fields in the logical and physical buffer with start field order characters and attributes bytes. The attribute bytes for all fields except the input area are set to 'protected; low-intensity.' The input area is unprotected. An insert-cursor order causes the cursor to appear at the beginning of the input area.

Extended Description	Module	Label
2 The GTSIZE macro determines terminal type. If the terminal is a 3270, the macro returns a value of 24 lines. If it does not, then issue a warning message (ERB204I) to the user, but allow him to proceed.		
3 Allocate the local 3270, using the session name as unit address, and open it.		
4		

Diagram 73. Picture Build (ERBPCTBL) (Part 1 of 2)

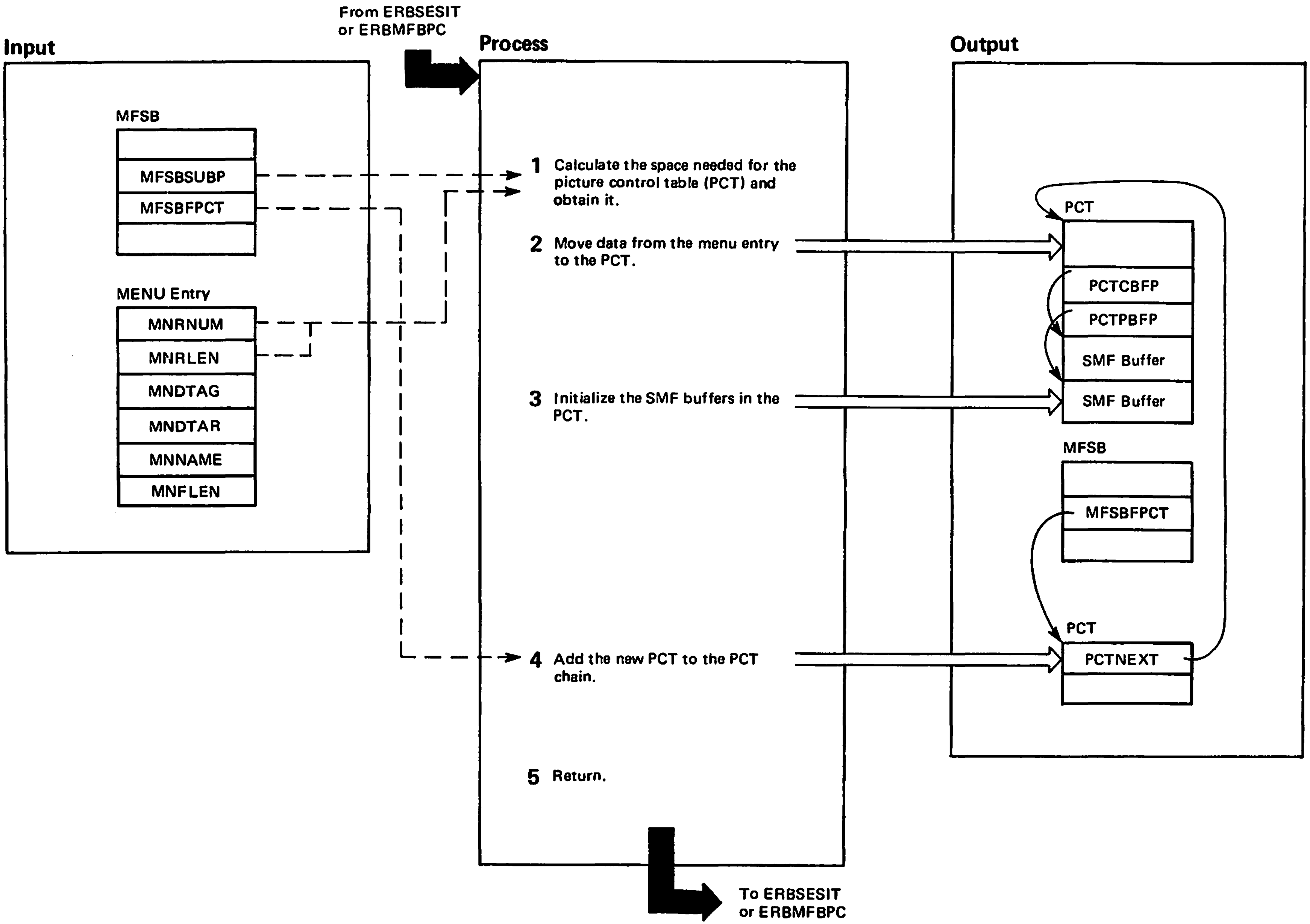


Diagram 73. Picture Build (ERBPCTBL) (Part 2 of 2)

Extended Description	Module	Label
Picture build creates and initializes the picture control table (PCT) in which all the data relating to a particular picture or measurement is kept. In the display monitor environment, there is one PCT for each menu entry. Session initialization (ERBSESIT) calls this module once for each menu entry. In the background monitor environment, this module is called once for each measurement specified in the input parameters. ERBMFBPC obtains the menu entry for each requested measurement and passes it to ERBPCTBL. After all the calls to ERBPCTBL are completed the resulting PCT chain represents the active measurements or pictures.		
1 The storage must be large enough to contain the base PCT and 2 SMF buffers. The size of the SMF buffer is the size of a fixed header plus the number of relocate blocks (MNRNUM) times the size of a relocate block (MNRLEN) plus the length of the Monitor II data control section (MNFLEN). If MNRNUM is 0, then the maximum number of address spaces (ASVTMAXU) is used instead. The SMF buffers are the means of communications between data gatherers and reporters. There are 2 buffers so they can be filled alternately and data calculations can be made.	ERBPCTBL	
2 As the default operands are moved to the PCT they are scanned to remove leading blanks and to set the length field to the actual number of characters in the buffer.		
3 Lengths and offsets in each SMF buffer are initialized.		
4 If there are no PCT's currently chained (MFSBFPCT = 0) then put the address of the new PCT in MFSBFPCT. If there are PCTs, follow the chain to the last one and set its PCTNEXT field to the address of the new PCT. The address of the new PCT is also put in MFSBCPCT for use by ERBMFBPC.		

Diagram 74. Display Process Control (ERBMFDPC) (Part 1 of 4)

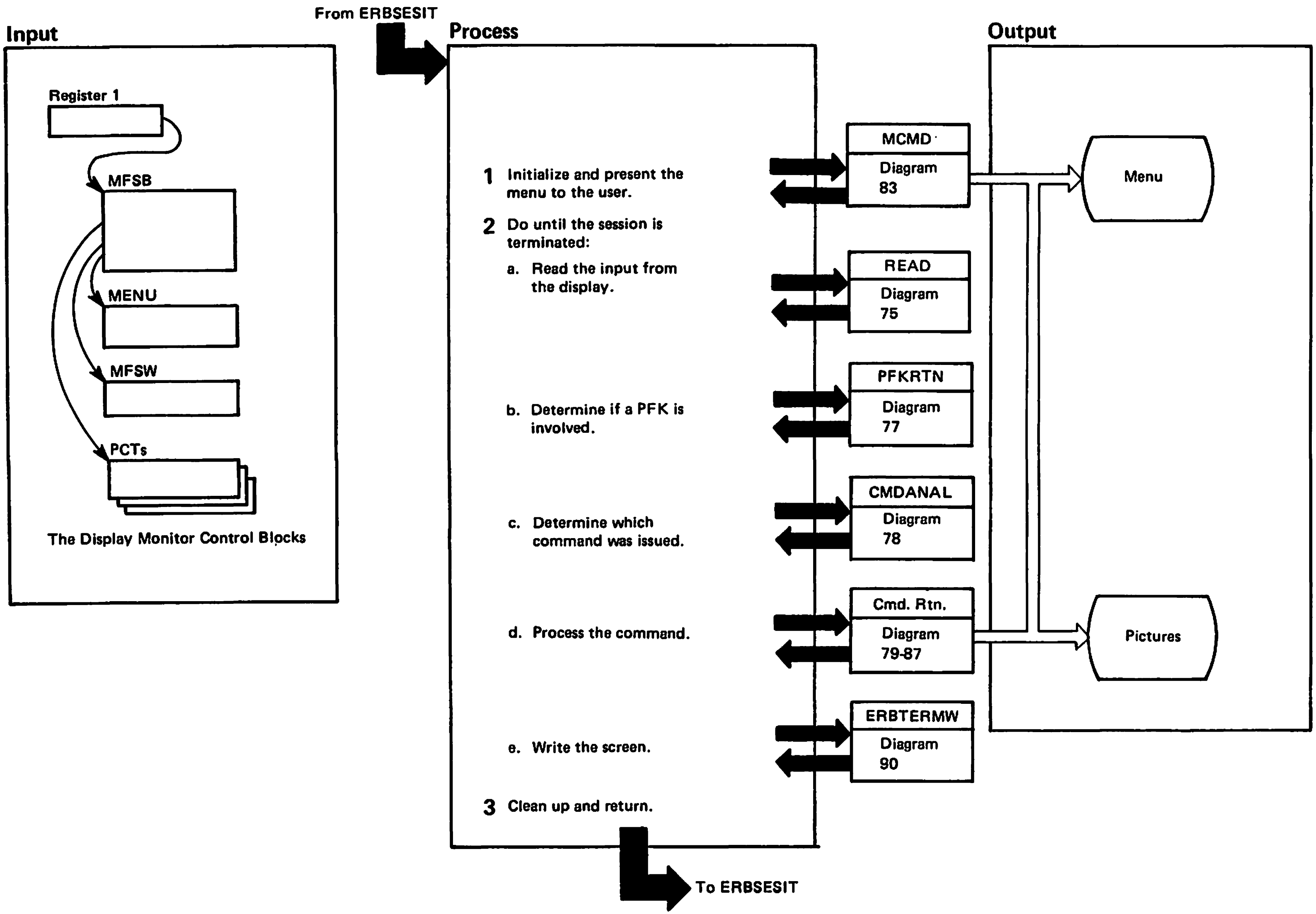


Diagram 74. Display Process Control (ERBMFDPC) (Part 2 of 4)

Extended Description	Module	Label
ERBMFDPC controls communication with the user at a display terminal and the execution of the pictures (display reports) requested.		

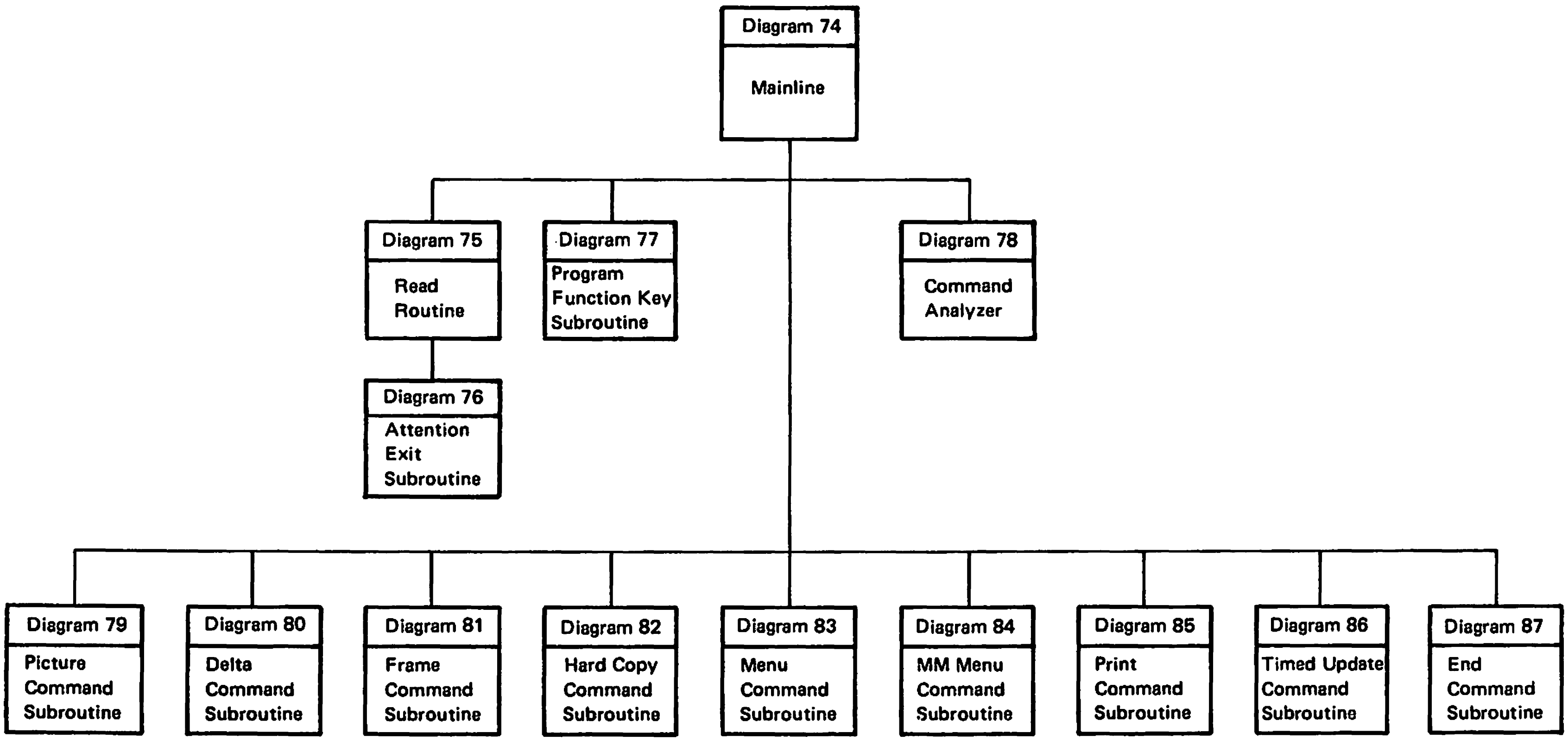


Diagram 74. Display Process Control (ERBMFDPC) (Part 3 of 4)

No diagram.

Extended description continued on next page.

Extended Description	Module	Label
1 Initialization includes priming the previous picture command buffer (MFSBPPCB) with the picture name from the first PCT. (This is done so that, if blank input is received before any valid pictures, the picture listed first in the menu is executed.) A pseudo PCT is built for the gatherer that creates the system status information.	ERBMFDPC	ERBMFDPC
Establish addressability to the session control block (MFSB) and the screen work area (MFSW). The MFSB contains session control variables required for session management and processing functions. The MFSW contains the BTAM DCB, the 3270 status bytes, the physical data area written to and read from the screen, and the logical data area for the data collected.		
Initialize an internal menu table; this table contains menu items and their associated PFK numbers from ERBFMENU, arranged in ascending order according to PFK number.		
Call the MCMD subroutine to format the menu and then call ERBTERMW (terminal write) to send the menu to the terminal. Set up buffers and work areas. Initialize a control block (MRCB) that the ESTAE can use to schedule a retry.	ERBMFDPC ERBTERMW	MCMD
2 Step 2 is the main control loop, which executes until the session terminates, either at the user's request or the operator's. When the session is to be terminated, MFSBTF (terminate flag) or MFSBETF (error terminate flag) will be on.		
a. Call the READ subroutine to obtain input from the display terminal and to handle errors, if any. Upon return, the input is in the MFSWITXT.		READ
b. Call the PFKRTN (Program Function Key) subroutine to find out if a PFK is involved. Upon return, the input is in VARTXT.		PFKRTN
c. If the request is a timed update, indicate this fact and the number of remaining intervals in the input area. If the request is not a timed update, blank the input area. In either case, call the CMDANAL subroutine to find the command in the buffer. The command analysis subroutine isolates the command and operands, if any, and decides which subroutine should process the command. This information is returned in an internal control block, the command analysis block (CAB).		CMDANAL

Extended Description	Module	Label
d. Call the subroutine whose address was returned by CMDANAL. These subroutines update the logical screen buffer in the MFSW. Format the status area according to MFSRCRID (current report name), MFSBDELF (delta/total flag), and MFSBHF (hardcopy flag). If neither menu is to be displayed, call ERBPUTSM (Put Stream) to put the status information in the logical screen buffer.	ERBPUTSM	
e. Call ERBTERMW to copy data from the logical screen buffer to the physical screen buffer and TPUT or WRITE it to the device.	ERBTERMW	
3 Cleanup consists of calling ERBMFALL to close the hardcopy data set if MFSBHOPN is on, indicating hardcopy had been opened. Delete the attention exit if the READ subroutine had established it.	ERBMFALL	
Return to the caller.		

Diagram 75. READ Subroutine (ERBMFDPC) (Part 1 of 2)

From ERBMFDPC Mainline

Input

MFSB
- MFSBENVC
- MFSBMECB
- MFSBSWAP

MFSW

NUMTIMES

WAITTIME

Process

1 If this is a TSO session and automatic screen (timed) update is requested, establish an attention exit.

2 Set the input buffer to blanks.

3 If this is a TSO session with no timed update requested, issue TGET to read the input and handle any errors.

4 If this is a local 3270 session with no timed update requested, then:
- Issue BTAM READ to get the input.
- Wait for the read to complete.

5 If timed update was requested, wait for the specified interval to elapse and delete the attention exit.

6 Return.

To ERBMFDPC Mainline

Output

MFSW
- MFSWTGBF

NUMTIMES

Diagram 75. READ Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Label
1 If this is a TSO session and the user has requested automatic screen update (timed update) then issue the STAX macro instruction to establish the attention exit, ATTNRTN (see Diagram 65).	ERBMFDPC	READ
2 Set the input buffer (MFSWTGBF) to blanks.		
3 This is a TSO session if the MFSBTSO bit in MFSBENVC is on. The TGET is issued with ASIS and WAIT options. The input area of the display is put in the buffer MFSWTGBF.		

Error handling:

RC = 4 — abend user 1401
RC = 8 — repeat TGET
RC = 12 — rewrite screen with message 'REENTER INPUT'
RC = 16 or greater — abend - user 1401

4 This is a local 3270 session if the MFSBFDM bit in MFSBENVC is on. After the BTAM READ, check for logical errors. If there are none, wait on 2 ECB's. The first will be posted when the READ completes, the other if the operator terminates the session.

- If the READ-complete ECB is posted, handle physical errors, if any, and return.
- If the stop-session ECB is posted, set the terminate flag (MFSBTF) so the session will terminate and return.

5 Automatic screen update (timed update) was requested and has been set up by the Timed Update Command Subroutine (Diagram 74). Decrease the count of the number of updates (NUMTIMES) by one. Set up the interval to elapse between this report and the next by issuing the STIMER macro instruction with the WAIT operand for the number of seconds the user specified on the T command (WAITTIME). When the STIMER wait completes, delete the attention exit. Note that the user can interrupt during the STIMER wait but not during report processing.

6 Return to the caller.

Diagram 76. ATTNRTN Subroutine (ERBMFDPC) (Part 1 of 2)

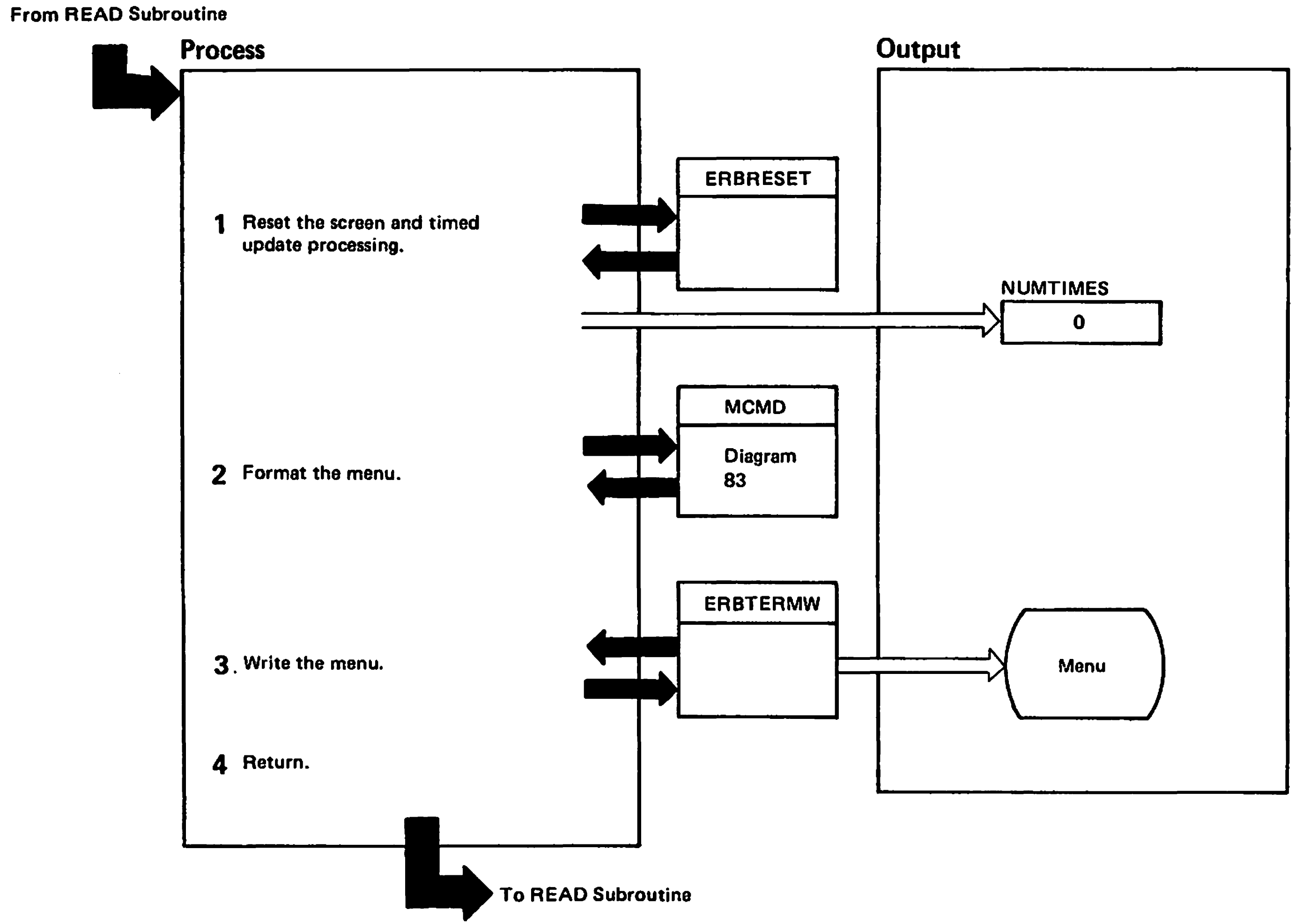


Diagram 76. ATTNRTN Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Label
The Attention Exit Subroutine (ATTNRTN) gets control when a user who requested timed update presses the attention key while the READ Subroutine is processing the timed update request. It can get control at any point during READ processing. It returns control to the point when the attention interruption occurred.		
1 Reset the number of updates (NUMTIMES) to zero. Call ERBRESET to reset all the screen indexes so the menu will start at the top of the logical buffer.	ERBMFDPC ERBRESET	ATTNRTN
2 Call the MCMD Subroutine to format the menu.		MCMD
3 Call ERBTERMW to send the menu to the terminal.	ERBTERMW	
4 Return control to attention interruption processing and eventually to the point in READ Subroutine processing where the attention interruption occurred. The READ Subroutine then processes the input the user entered after generating the attention interruption. Thus, processing of a new command begins.		

Diagram 77. PFKRTN Subroutine (ERBMFDPC) (Part 1 of 2)

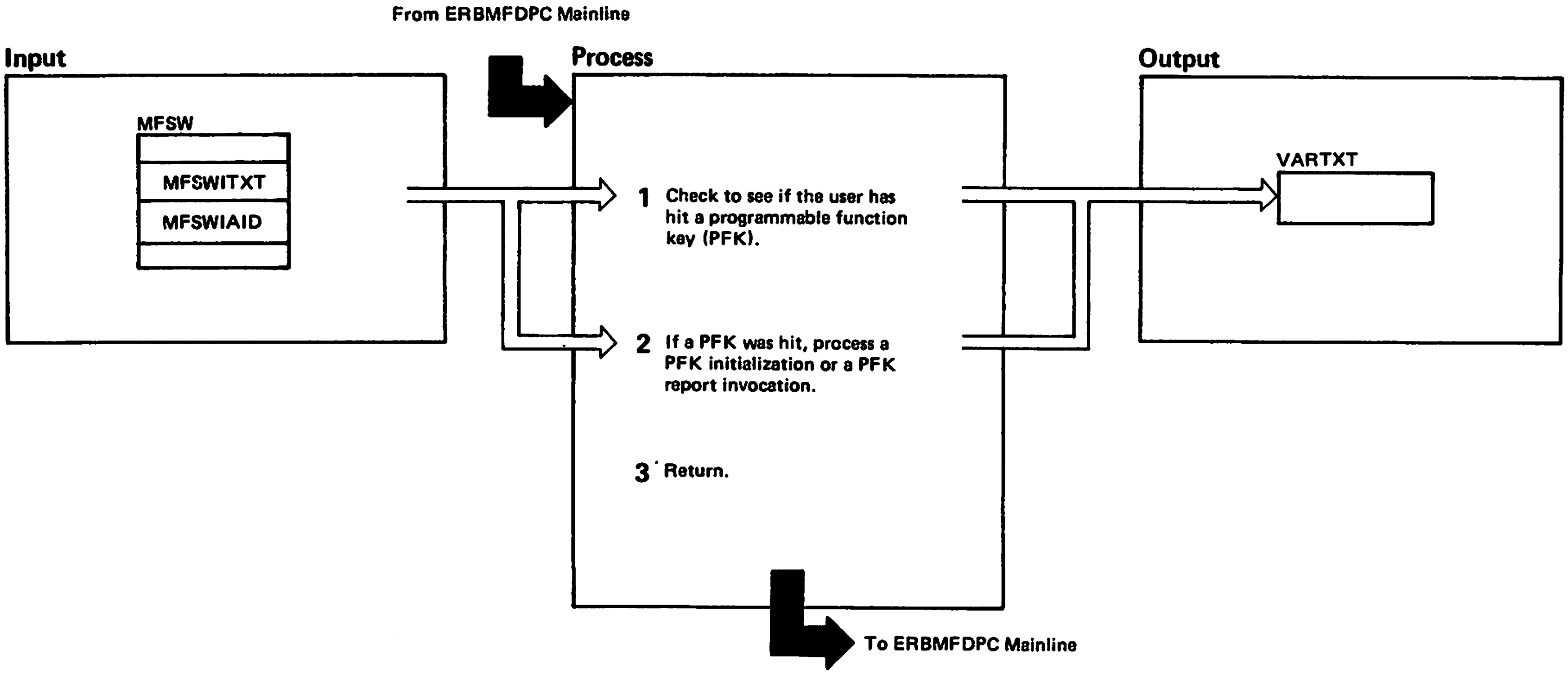


Diagram 77. PFKRTN Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Label
The PFKRTN Subroutine determines if the user has hit a PFK. If a PFK is involved, the routine processes either the initialization of a PFK or the report invocation by means of a PFK.		
1 If the user hit the enter key, prime VARTXT (the input area for the CMDANAL Subroutine) with the input entered. Return to the mainline routine.	ERBMFDPC	PFKRTN
If the user hit the PA2 key (the MENU key), prime VARTXT with the menu command. Return to the mainline routine.		
If neither the enter key nor PA2 was used, determine if the user hit a valid PFK. If not, prime VARTXT with the menu command to display the menu. Return to the mainline routine.		
2 A valid PFK was hit. If the first non-blank character in the input area (MFSWITXT) is a pound sign (#), then the PFK hit is being initialized to correspond to the data the user has placed in the input area (assumed to be a menu item and optional operands). Store the menu item name, any operands specified, and the associated PFK number in the internal menu table. The PFK is now initialized for the duration of the session. Each subsequent use of the PFK invokes the stored menu item and any associated operands. If the menu item name exceeds the maximum possible command length, set a return code of 4 and call the error subroutine (DGDRERR) to produce an error message.		
If the first non-blank character is not a pound sign (#), then the PFK hit is being used to invoke a report. Use the internal menu table to find the menu item and operands associated with the PFK. Prime VARTXT with this menu item and operands.		
3 Return to the caller.		
Error Processing performed by DGDRERR		
4 'INVALID OPERAND SYNTAX' in message area.		

Diagram 78. CMDANAL Subroutine (ERBMFDPC) (Part 1 of 2)

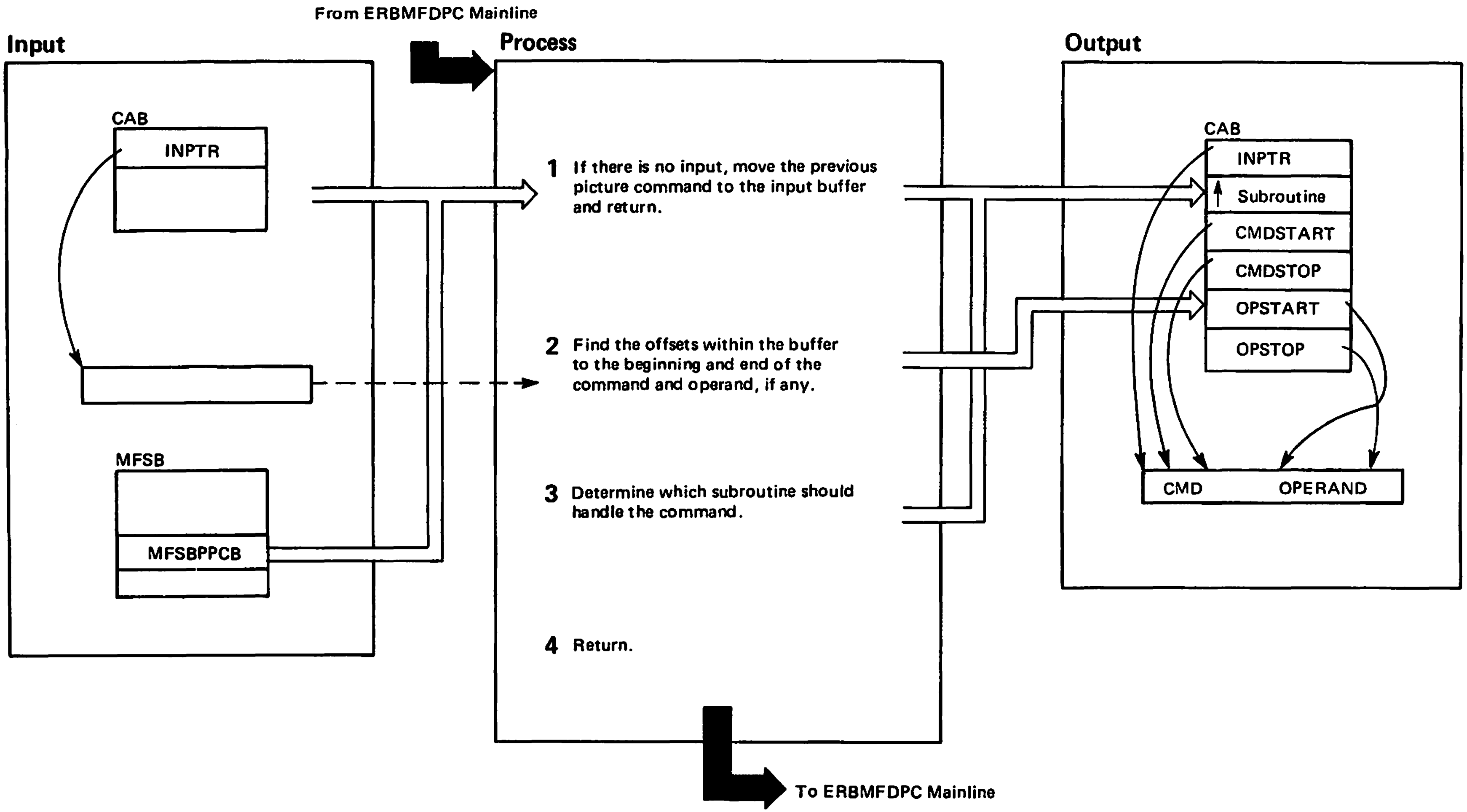


Diagram 78. CMDANAL Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description

Module **Subroutine**

ERBMFDPC CMDANAL

1 The move is done so that the effect of hitting 'ENTER' with no input is to reexecute the last picture command (saved in MFSBPPCB.)

2 The beginning offsets are found by scanning for the next non-blank. The ending offsets are found by scanning for the next blank.

3 The subroutine is determined according to the following table. Its address is stored in the CAB.

Command	*Command Subroutine*	*See Diagram Number*
D	DCMD	80
F	FCMD	81
H	HCMD	82
M	MCMD	83
MM	MMCMD	84
P	PCMD	85
T	TIMED	86
Z	ZCMD	87
Other	PCCMD	79

Diagram 79. PCCMD Subroutine (ERBMFDPC) (Part 1 of 2)

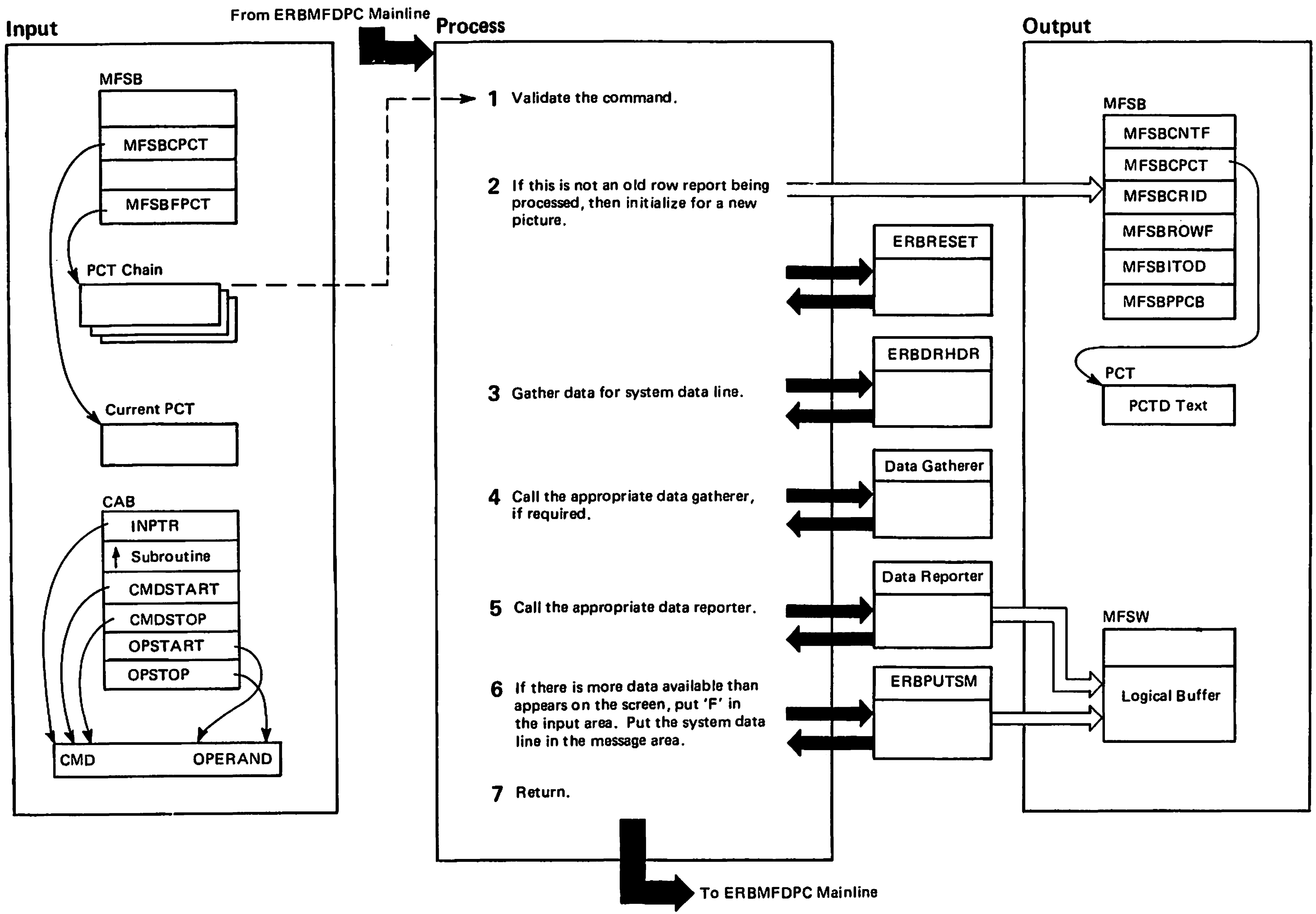


Diagram 79. PCCMD Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Subroutine
1 Handle a Recall request by checking the first character of the command for 'R'. If it is 'R', then adjust the CMDSTART value past it and cause the data gatherer call to be bypassed by setting DGFLAG off. Validate the command by comparing it to the PCTNAME field in each PCT in the chain. If it doesn't match any of them, put out the invalid command message and return. If it does match, keep the address of the *new* PCT.	ERBMFDPC	PCCMD

2 This is an old row report if:

- MFSBROWF is on,
- the new PCT and current PCT are the same (MFSBCPCT), and
- the operands match.

Externally, the user requested the second (or later) data line of a row report.
If the above test fails, initialize for a new picture by

- filling in the following fields:
  MFSBCNTF, MFSBCPCT, MFSBCRID, MFSBROWF, MFSBITOD, MFSBPPCB, PCTOTEXT.
- calling ERBRESET to reset the screen and, if necessary, to put report delimiters and time delimiters to hardcopy.

3 If DGFLAG is on, gather data for the system data line by calling ERBDRHDR using the pseudo PCT. Copy the data returned into the current PCT. This data can then be associated with the data to be put in the current SMF buffer in case it is later recalled.

4 If DGFLAG is on, call the data gatherer. PCTDGP contains the entry point of the gatherer. If it is 0, this is the first time this gatherer was used, so LOAD it using the name in PCTDGNM and store the entry point at PCTDGP. If the gatherers returns a non-zero return code, set DRFLAG off to prevent the data reporter call and call the error subroutine (DGDRERR).

Extended Description	Module	Subroutine

5 If DRFLAG is on, call the data reporters. PCTDRP contains the entry point of the reporter. If it is 0, this is the first time this reporter was used, so LOAD it using the name in PCTDRNM and store the entry point at PCTDRP. If this is a new row report, call the reporter with entry code = 1 to put out the report headers.

Call the reporter with entry code = 2 to put out one line of data (row report) or headers and full set of data (table report). On this call indicate in the calling sequence either delta or total values are required. See Diagram 80.

If there is a non-zero return code from either call, call the error subroutine (DGDRERR).

6 If the gatherer and reporter completed successfully, finish the process by:

- putting an 'F' in the input area to indicate that there is more data available than appears on the screen (if that is the case).
- copying the system data line from the current PCT to the message area of the display (via ERBPUTSM).

7 Return to caller.

Error Processing performed by DGDRERR

Return Code Value	*Action*
4	'INVALID OPERAND SYNTAX' in message area
8	user abend 1402
12	msg ERB403I and ERB404I
16	msg ERB405I
20	msg ERB406I
24	msg ERB407I
>24	msg ERB408I

Diagram 80. DCMD Subroutine (ERBMFDPC) (Part 1 of 2)

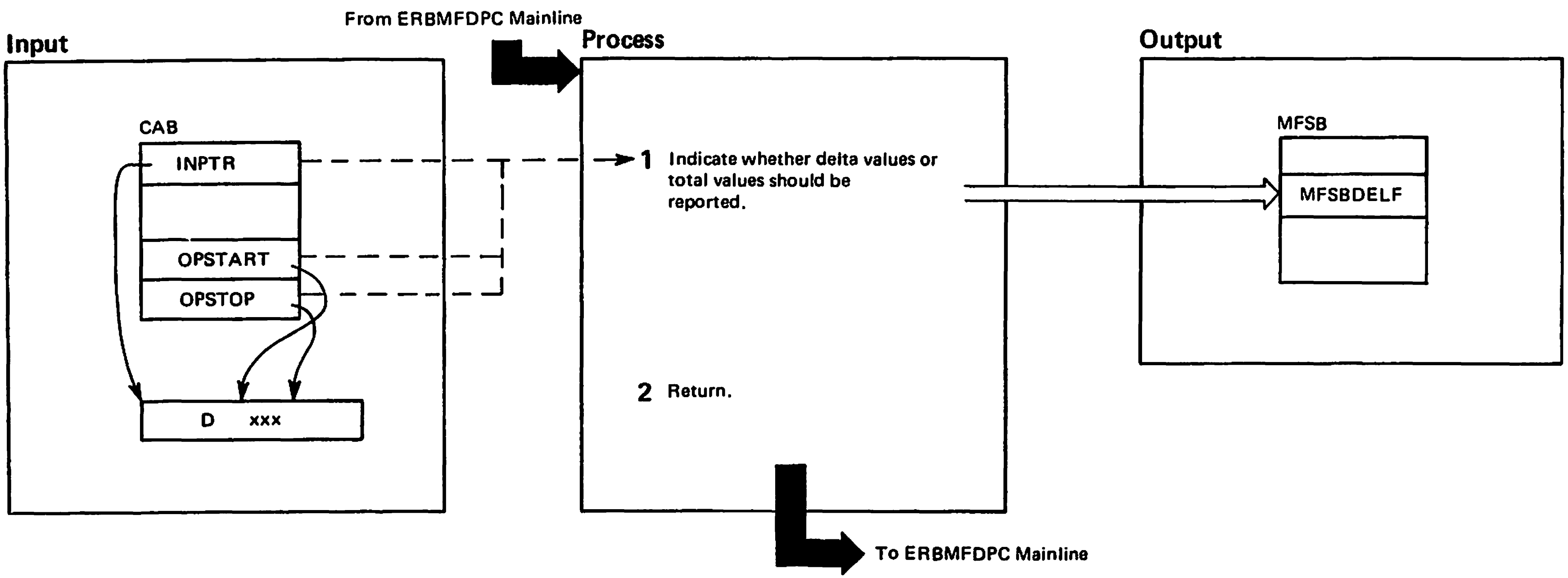


Diagram 80. DCMD Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Subroutine
The DCMD subroutine gets control when the user enters 'D', the delta command. 'D ON' indicates that the reporters should report delta values, the difference between the values now and the last time they were reported. 'D OFF' indicates that total values should be reported. 'ON' is the default.	ERBMFDPC	DCMD

1 This subroutine simply sets the MFSBDELF bit according to the operand. PCCMD subroutine indicates the value of the bit to the reporters it calls. The state of the bit is reflected in the status area of the screen as 'D' or 'T'. This area is formatted by the main processing loop after this subroutine returns for it.

2

Diagram 81. FCMD Subroutine (ERBMFDPC) (Part 1 of 2)

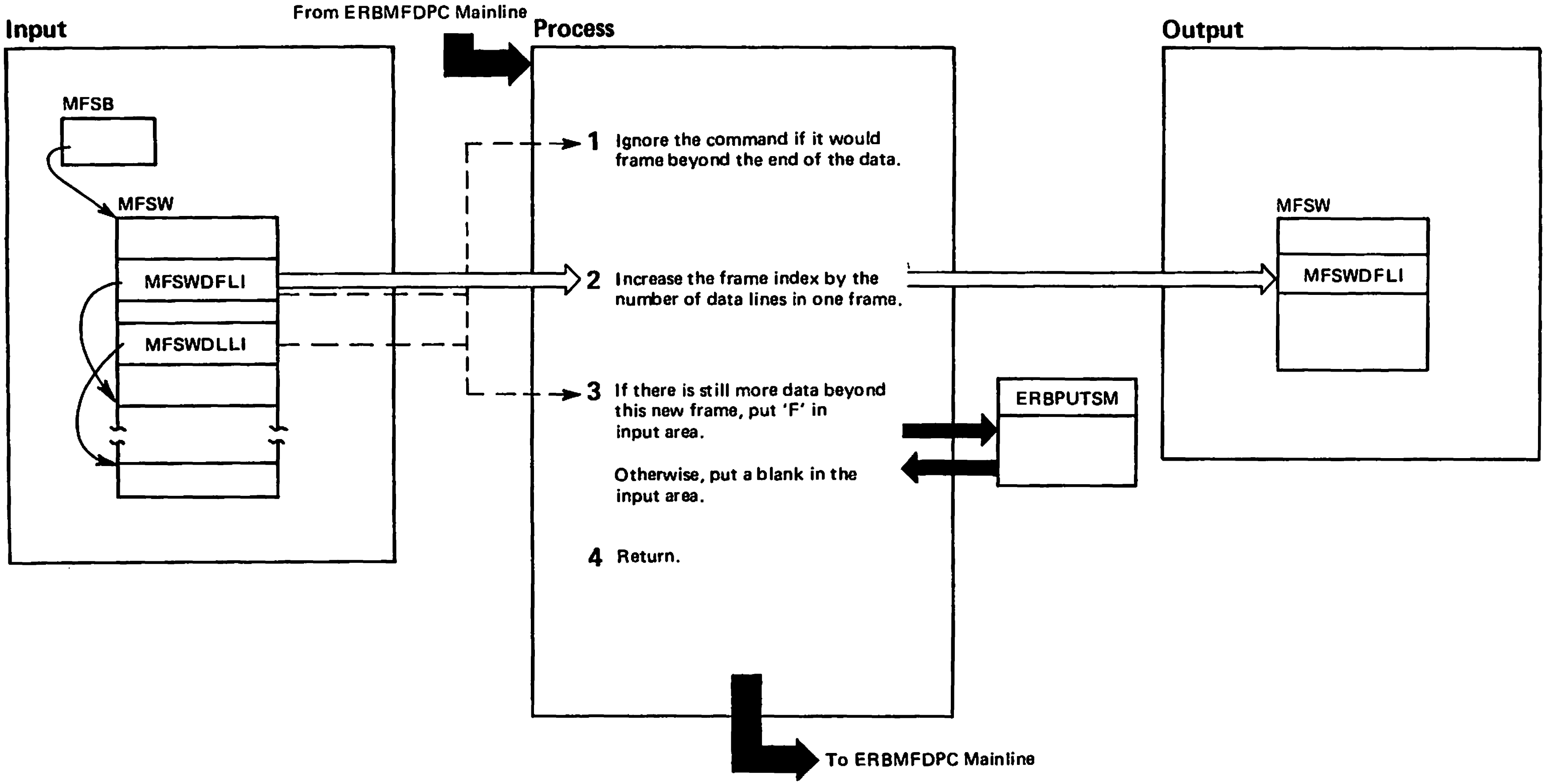


Diagram 81. FCMD Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Subroutine
1 If this command would cause a frame beyond the end of data by (compare it against MFSWDLLI, the index to the last data line), go to step 4 and return.		
2 Increase MFSWDFLI (current frame index) by 21, the number of data lines in one frame. MFSWDFLI tells ERBTERMW where to start moving data lines from the logical buffer to the physical buffer. Thus, when it is called by the main loop after this subroutine completes, the next frame of data will be transmitted to the device.		
3 Compare the current MFSWDFLI and 21 with MFSWDLLI to see if there is still more data in the logical buffer that the user has not seen.	ERBPUTSM	
4		

Diagram 82. HCMD Subroutine (ERBMFDPC) (Part 1 of 2)

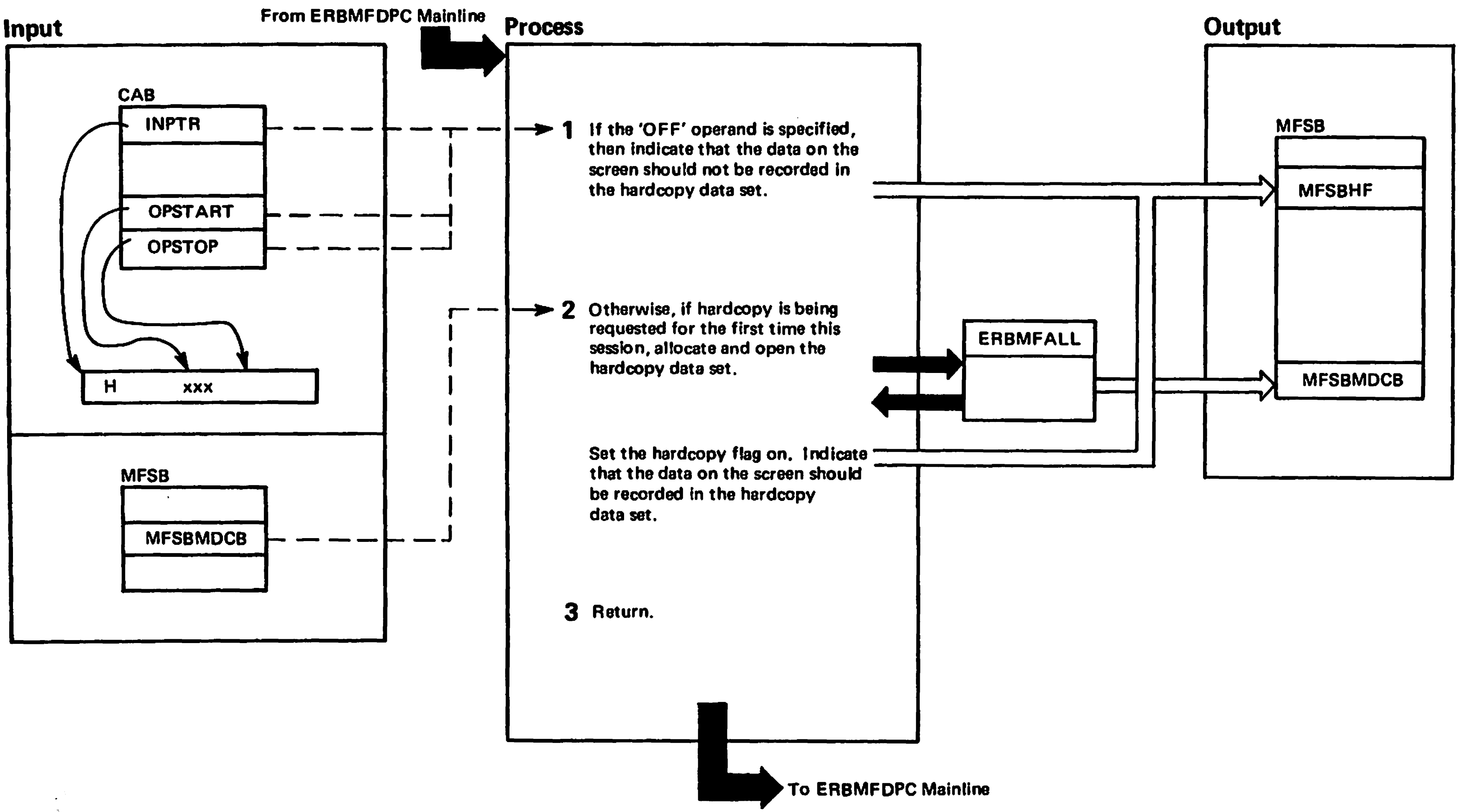


Diagram 82. HCMD Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Subroutine
1 The HCMD subroutine gets control when the user enters 'H', the hardcopy command. 'H ON' indicates that the data that is to appear on the screen should also be recorded in the hardcopy data set. This routine sets MFSBHF, the hardcopy flag (when called by a reporter) and if on, then sends the data to hardcopy. ERBRMFPL checks the flag.	ERBMFDPC	HCMD
2 If this is the first time hardcopy is used, MFSBMDCB, the pointer to the hardcopy DCB, will be 0. If that is the case, call ERBMFALL to allocate, open and put headers to the data set.	ERBMFALL	
The data set is not closed or unallocated when H OFF is specified so that the user can turn hardcopy on and off several times during the session and end up with one output data set. The print command also uses this data set.		
3		

Diagram 83. MCMD Subroutine (ERBMFDPC) (Part 1 of 2)

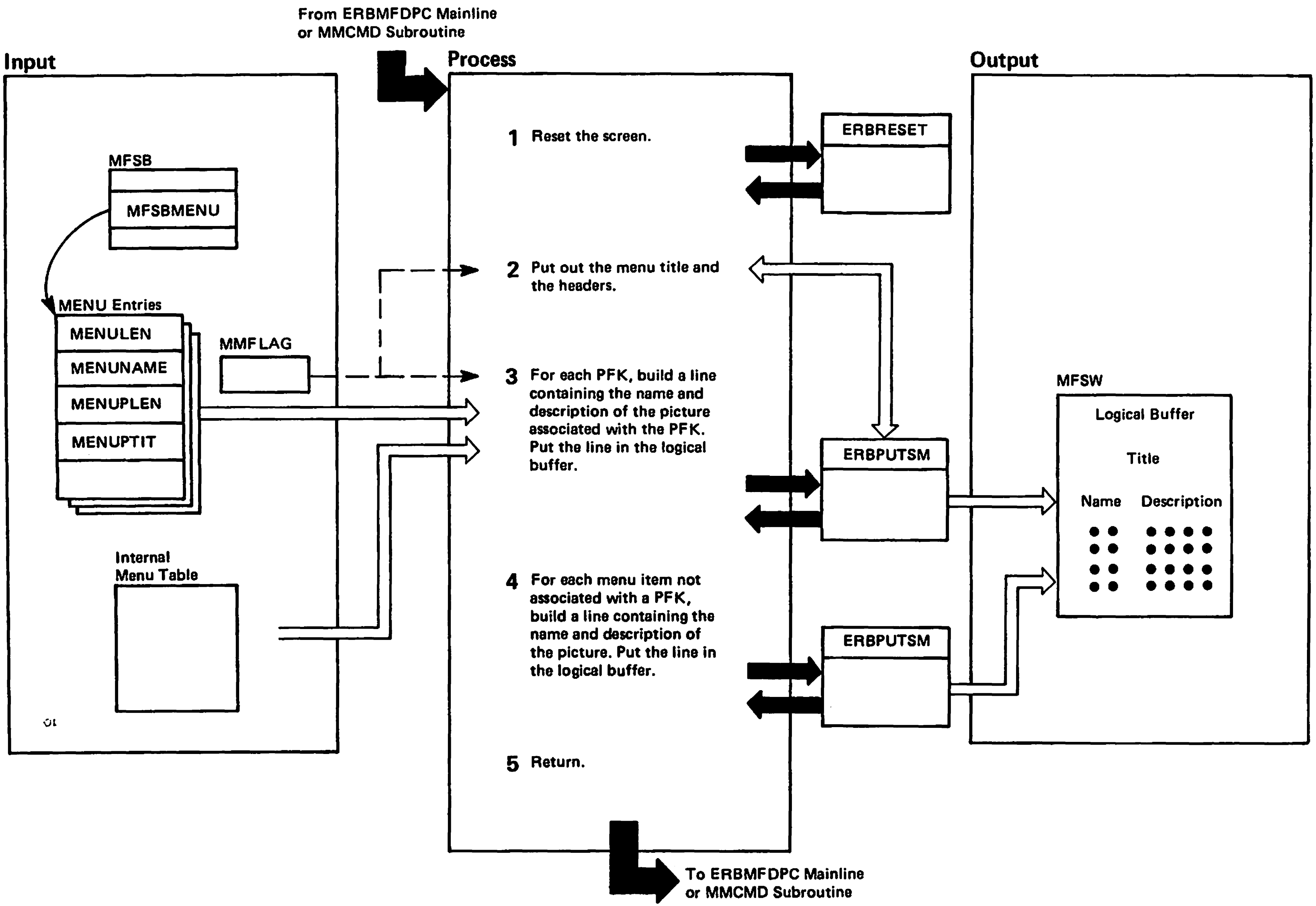


Diagram 83. MCMD Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Label
This subroutine formats either the display menu or the default operands menu. It is called during the initialization of the session to present the display menu to the user. It is invoked later during the session from ERBMFDPC mainline when the user enters the 'M' command to see the menu again or from the MMCMD Subroutine when the user enters the 'MM' command to see the default operands menu.	ERBMFALL	MCMD
1 Call ERBRESET to reset all the screen indexes so the menu will start at the top of the logical buffer.	ERBRESET	
2 Call ERBPUTSM to put the menu title and the headers for the name and description columns to the logical buffer. If MMFLAG is off, the headers define the display menu. If MMFLAG is on, the headers define the default operands menu.	ERBPUTSM	
3 Use the internal menu table to build a line for each PFK. The internal menu table contains an entry for each PFK that has been assigned, arranged in ascending order according to PFK number. For each entry, use the picture name to search ERBFMENU to find a matching entry. If a match is found in ERBFMENU, build a line containing the picture name and the associated PFK number. If 'M' was entered (MMFLAG is off), then also include the report description. If 'MM' was entered (MMFLAG is on), then include the gatherer and reporter operands, if any. (Include the reporter operands only when they differ from the gatherer operands.) If no match is found in ERBFMENU, then the PFK is not associated with a picture name but with a display command. Build a line containing the PFK number, the command, and the operands associated with it.		
Call ERBPUTSM to put the line to the logical buffer.	ERBPUTSM	

Extended Description	Module	Label
4 Build a line for each entry in ERBFMENU that is not associated with a PFK and thus was not listed before (in step 3). If MMFLAG is off, then build a line that contains the picture name and the report description. If MMFLAG is on, then build a line that contains the gatherer and reporter operands, if any. (Include the reporter operands only when they differ from the gatherer operands.)		
Call ERBPUTSM to put the line to the logical buffer.	ERBPUTSM	
5 The menu is now formatted in the logical buffer. Return to the main processing loop where it will be put to the terminal or to the MMCMD Subroutine.		

Diagram 84. MMCMD Subroutine (ERBMFDPC) (Part 1 of 2)

From ERBMFDPC Mainline

Process

1 Indicate that the default operands menu was requested.

2 Format the menu.

3 Return.

To ERBMFDPC Mainline

MCMD

Diagram 83

Output

MMFLAG

Diagram 84. MMCMD Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Label
This subroutine is called when the user enters the 'MM' command to request a display of the default operands.		
1 Turn MMFLAG on (MMFLAG=1) to indicate that the user requested the default operands menu.	ERBMFDPC	MMCMD
2 Call the Menu Command (MCMD) Subroutine to format the menu in the logical screen buffer. Because MMFLAG is on, the MCMD subroutine will format the default operands menu.		MCMD
3 The default operands menu is now formatted in the logical buffer. Return to the main processing loop where it will be put to the terminal.		

Diagram 85. PCMD Subroutine (ERBMFDPC) (Part 1 of 2)

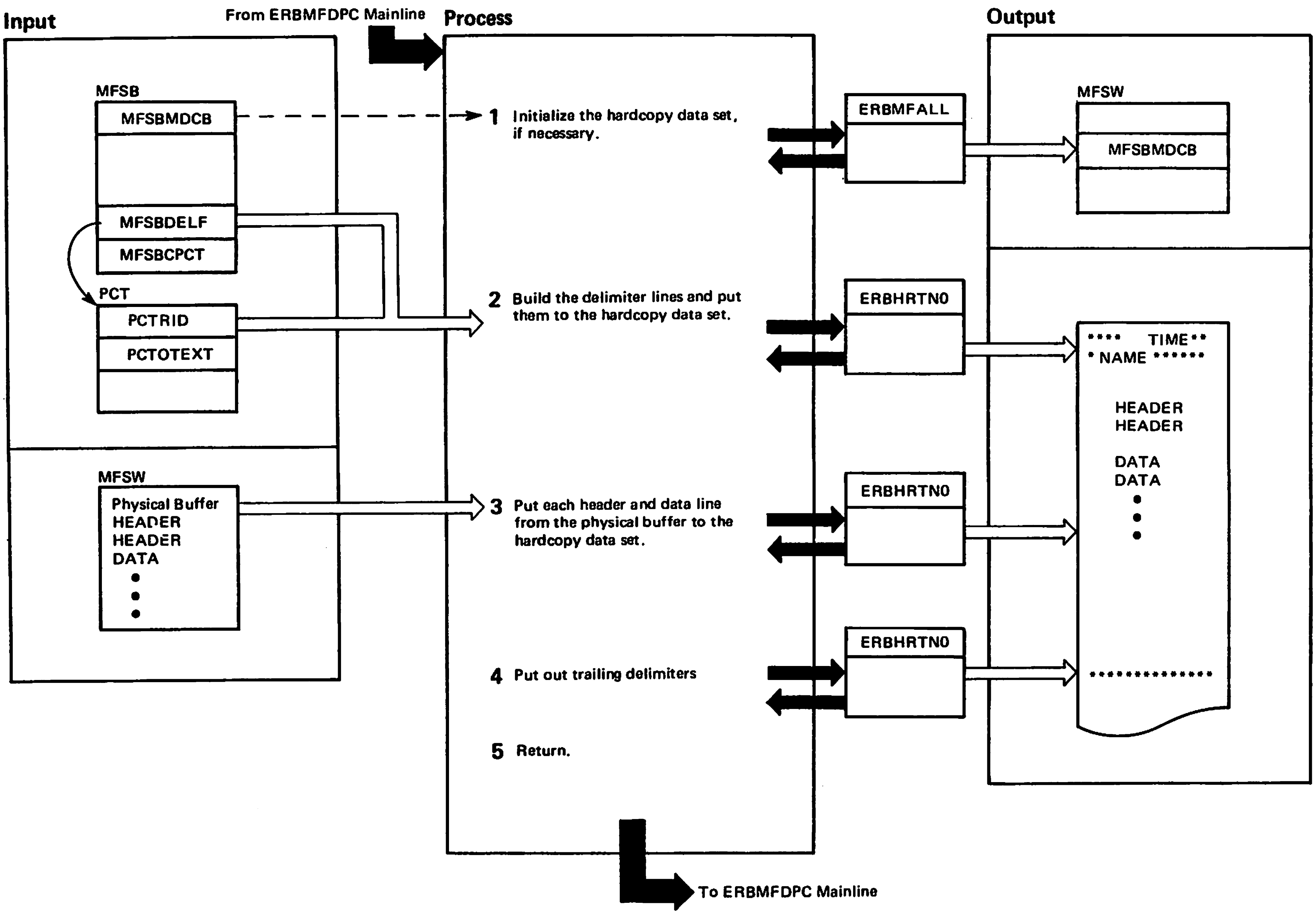


Diagram 85. PCMD Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Subroutine
This subroutine is called when the user enters the 'P' command to obtain a hardcopy of the data that appears on the screen.	ERBMFDPC	PCMD
1 The hardcopy function and the print command use the same data set. The first occurance of either 'P' or '4' allocates and opens the data set and it remains open until the session terminates. The data set needs to be allocated when MFSBMDCB is zero. After the data set is initialized MFSBMDCB points to DCB.	ERBMFALL	
2 The beginning delimiter lines contains time, data picture name and operands from the current PCT (Picture Control Table), the contents of the message area, and an indication of DELTA or TOTAL made from MFSBDELF.	ERBHRTNO	
3 Two header lines and 21 data lines are taken from the physical screen buffer MFSWPSBF and put to the data set by ERBHRTNO. A return code of 4 from a call to ERBHRTNO indicates an I/O SYNAD error occurred. If it occurred, message ERB403I is put out and this subroutine returns to the caller.	ERBHRTNO	
4		

Diagram 86. TIMED SUbroutine (ERBMFDPC) (Part 1 of 2)

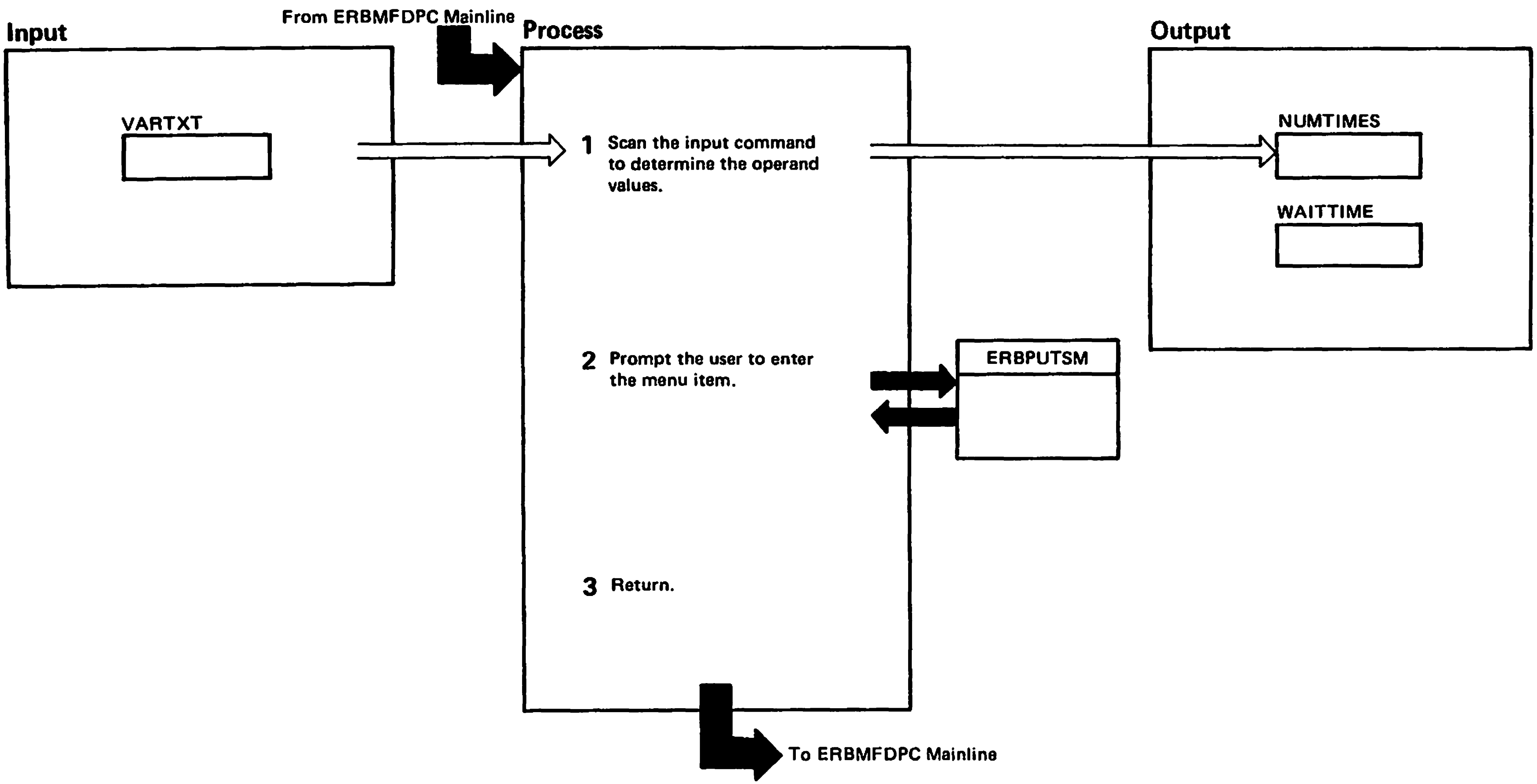


Diagram 86. TIMED SUbroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Label
The TIMED Subroutine processes the T command used to request automatic timed updates of a report.	ERBMFDPC	TIMED
1 Scan the operand (VARTXT) to determine the number of times the report is to be repeated (the first operand) and the number of seconds to elapse between updates of the report (the second operand). Set NUMTIMES equal to the number of times the report is to be repeated. Set WAITTIME equal to the number of seconds that are to elapse between reports. If the syntax is invalid, set a return code of 4 and call the error subroutine (DGDRERR) to produce an errror message.		
2 Prompt the user to enter the report to be repeated; call ERBPUTSM to place the prompting message in the logical screen area.	ERBPUTSM	
3 Return to the caller.		
Error Processing performed by DGDRERR		
4 'INVALID OPERAND SYNTAX' in message area.		

Diagram 87. ZCMD Subroutine (ERBMFDPC) (Part 1 of 2)

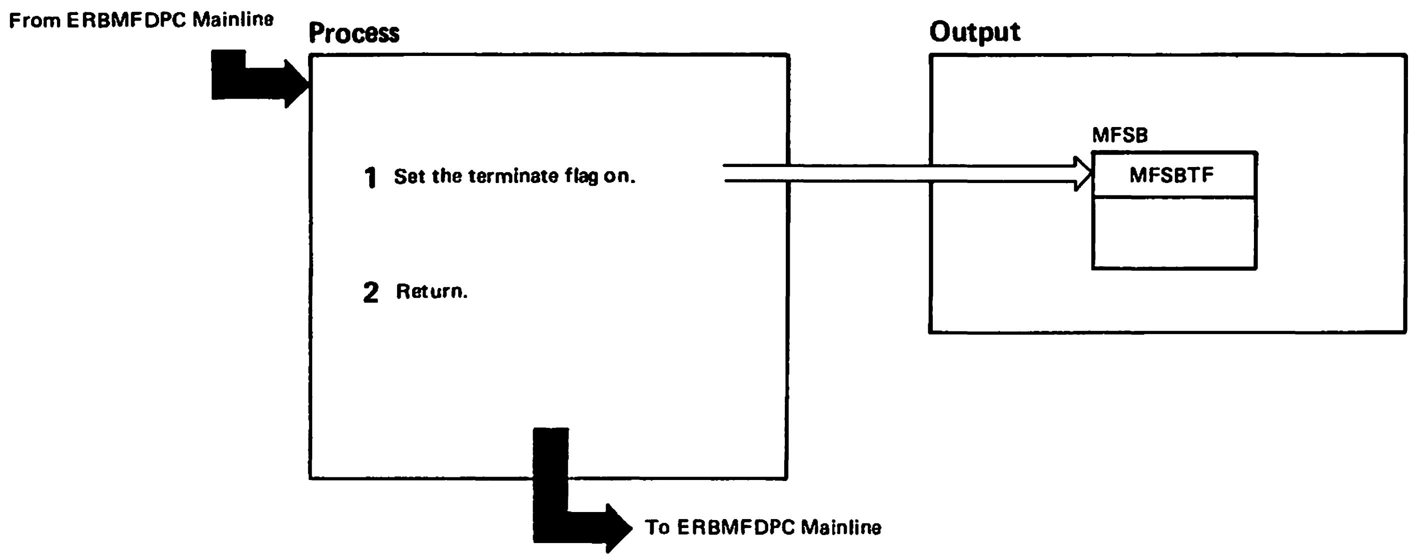
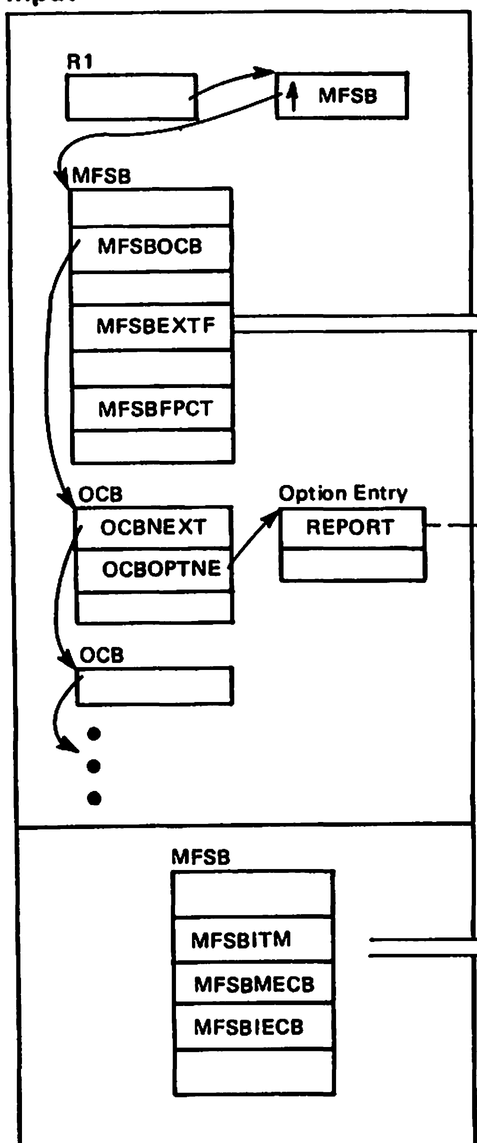


Diagram 87. ZCMD Subroutine (ERBMFDPC) (Part 2 of 2)

Extended Description	Module	Subroutine
1 When MFSBTF (the terminate flag) is on, the main processing loop terminates.		
2 Return to ERBMFDPC Mainline.		

Diagram 88. Background Process Control (ERBMFBPC) (Part 1 of 4)

Input



ERBSESIT

Process

Set Up to Gather Data

1 Initialize the fields for the report options.

2 For the specified report options, update the appropriate fields in the MFSB. For each report a Picture Control Table (PCT) is built and the appropriate data gatherer is loaded.

ERBPCTBL

A 3 For each printed report requested:

B
- Load the appropriate data reporter.
- Set up the SYSOUT data set.
- Write the report delimiter to SYSOUT.

ERBMFALL

ERBRESET

Gather and Report the Data

4 For the requested number of intervals:

- Gather and report the data.
- Start the measurement interval and wait for it to complete or for the session to be modified. Go to step 6 if the session is modified.

ERBBDATA

Output

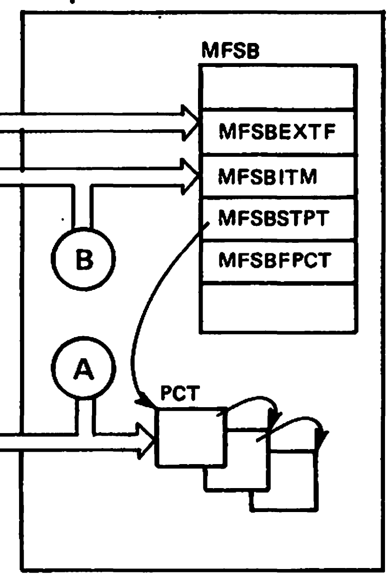


Diagram 88. Background Process Control (ERBMFBPC) (Part 2 of 4)

Extended Description	Module	Label

ERBMFBPC handles background Monitor II measurement collection.

1 Set MFSBEXTF to zero, which sets the record, report, and delta option flags and the row report flag off. Turn on the hardcopy flag (MFSBHF).

2 If 'DELTA' is specified, turn MFSBDELF on. If 'RECORD' is specified, turn MFSBSMFF on. If 'REPORT' is specified, turn MFSBRPTF on, and set MFSBRPTM according to whether 'DEFER' or 'REALTIME' is requested. Copy the SINTV value (the measurement interval value) from the OCB to MFSBITM. Move the STOP time to MFSBSTPT and calculate the number of intervals to stop time. For each report request (a positive option that's in the background menu), call ERBPCTBL to build a PCT, and then load the data gatherer.

3

4
- ERBBDATA calls the data gatherer, and if requested, writes the SMF record and/or calls the data reporter.
- Issue the STIMER macro instruction for the length of time specified in MFSBITM. (When the time interval is finished, control will be given to a routine which will post the interval timer ECB, MFSBIECB.) Wait for either the modify ECB (MFSBMECB), or MFSBIECB to be posted.

Diagram 88. Background Process Control (ERBMFBPC) (Part 3 of 4)

Input

MFSB
- MFSBMECB
- MFSBTF
- MFSBOCB

OCB

Process

5 If the requested number of intervals is completed, go to step 7.

6 If an interval ends with a request to modify a session:

- Collect and report the data for that interval.
- Update the report option fields.
- Rebuild the PCT chain and load any required new data gatherers.
- For each new printed report requested, if necessary:
  - – Load the appropriate data reporter.
  - – Set up a SYSOUT data set and write the report delimiter to SYSOUT.
- Indicate that the modifications are complete and go to step 4.

7 Close all Data Control Blocks.

8 Return.

ERBBDATA

ERBPCTBL

ERBMFALL

ERBRESET

ERBSESIT

Output

MFSB
- MFSBEXTF
- MFSBITM
- MFSBSTPT
- MFSBPCT
- MFSBMODC

PCT

Extended Description	Module	Label
5 If the MFSBIECB is posted, check whether the number of intervals requested has occurred. If it has, go to step 7; if not, go to step 4.		
6 If MFSBMECB was posted, check whether MFSBTF is on, indicating a request to terminate. If MFSBTF is on, go to step 7. If the session is continuing, but being modified: • Collect data for last interval. • Scan the OCB chain. Process DELTA, RECORD, SINTV and STOP options as before (see number 2). If the REPORT option or SYSOUT class was modified, close all SYSOUT DCBs (unless the change was from NOREPORT to REPORT). Set MFSBRPTF and MFSBRPTM. • Call ERBPCTBL for each new report request, and load the data gatherers. Delete PCTs and data gatherers and reporters for any reports that are no longer requested. If the operands for a report have been modified, move the new operand text from the OCB to the PCT. Reset PCTHPRT to 0 so that headers will be rewritten. If the printed report is continuing (REPORT flag is on and the SYSOUT DCB address is not 0), call ERBRESET to write the report delimiter showing new operands. • If the 'REPORT' option was requested (MFSBRPTF is on) then for each PCT: Turn on the report flag (PCTRPRT) and load the data reporter if its address (PCTDRP) is zero. If the sysout DCB address (PCTSDCB) is zero, indicating that either this is a new printed report request, or that the DCB was closed during prior modify processing, then call ERBMFALL to allocate, open, and write the header to the SYSOUT data set and call ERBRESET to write the report delimiter to the SYSOUT data set. • Post MFSBMODC to tell ERBMFCTL modify is complete.		

Extended Description	Module	Label
7		
8		

Diagram 89. Dynamic Allocation (ERBMFALL) (Part 1 of 2)

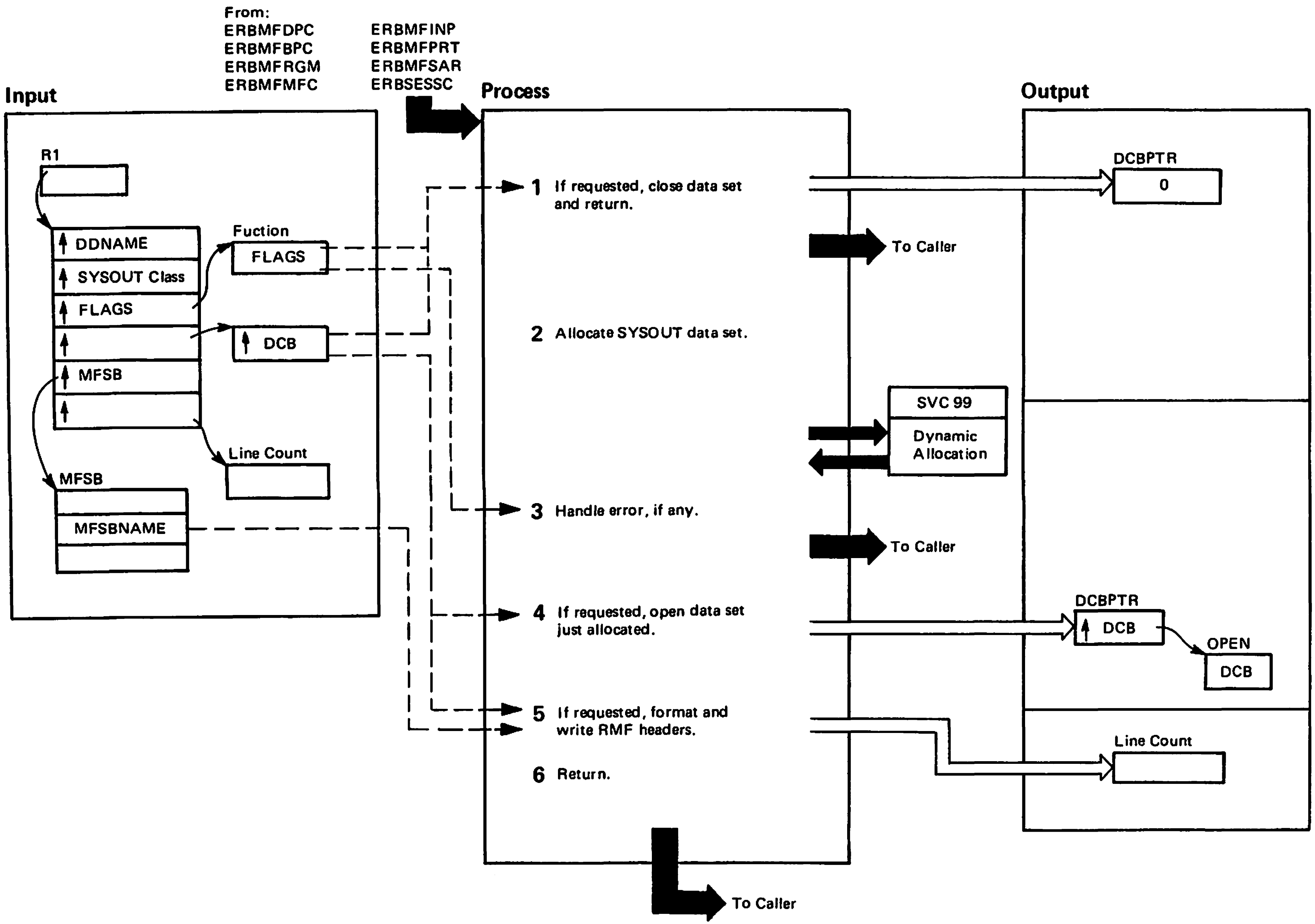


Diagram 89. Dynamic Allocation (ERBMFALL) (Part 2 of 2)

Extended Description	Module	Subroutine

Introduction:
ERBMFALL is called by many RMF modules to allocate their output data sets. Monitor II modules also request this module to OPEN the data set and put the standard RMF headers into it. Monitor II modules use ERBMFALL to CLOSE the data set after processing has completed.

1 If the CLOSE function flag is set, CLOSE the DCB pointed to by the fourth input parameter. Free the storage occupied by the DCB and set the pointer, passed by the caller, to 0.

2 Use the DDNAME and SYSOUT class provided by the caller. If requested in the function flags, use the "unallocate at close" key.

3 If the dynamic allocation failed due to "ddname in use" (error code = 410x), treat as success. (This allows an installation to preallocate data sets via JCL or the TSO ALLOCATE command specifying options other than those in this module.) If allocation is not successful, check function flags. If requested by function flags, issue message ERB232I including return code, error code and information code and return to caller with return code = 4. Otherwise issue user abend 1203.

4 If requested in the function flags, GETMAIN DCB, OPEN it and return DCB address in area provided by caller. An open exit fills in block size, if necessary, for preallocated data sets. The synad exit specified to OPEN is contained in this module.

5 If requested in the function flags, format and write RMF headers. In general, they follow the format of the headers created by ERBMFRGM. If requested, the total number of lines occupied by the headers is returned to the caller.

6

Diagram 90. Terminal Write (ERBTERMW) (Part 1 of 2)

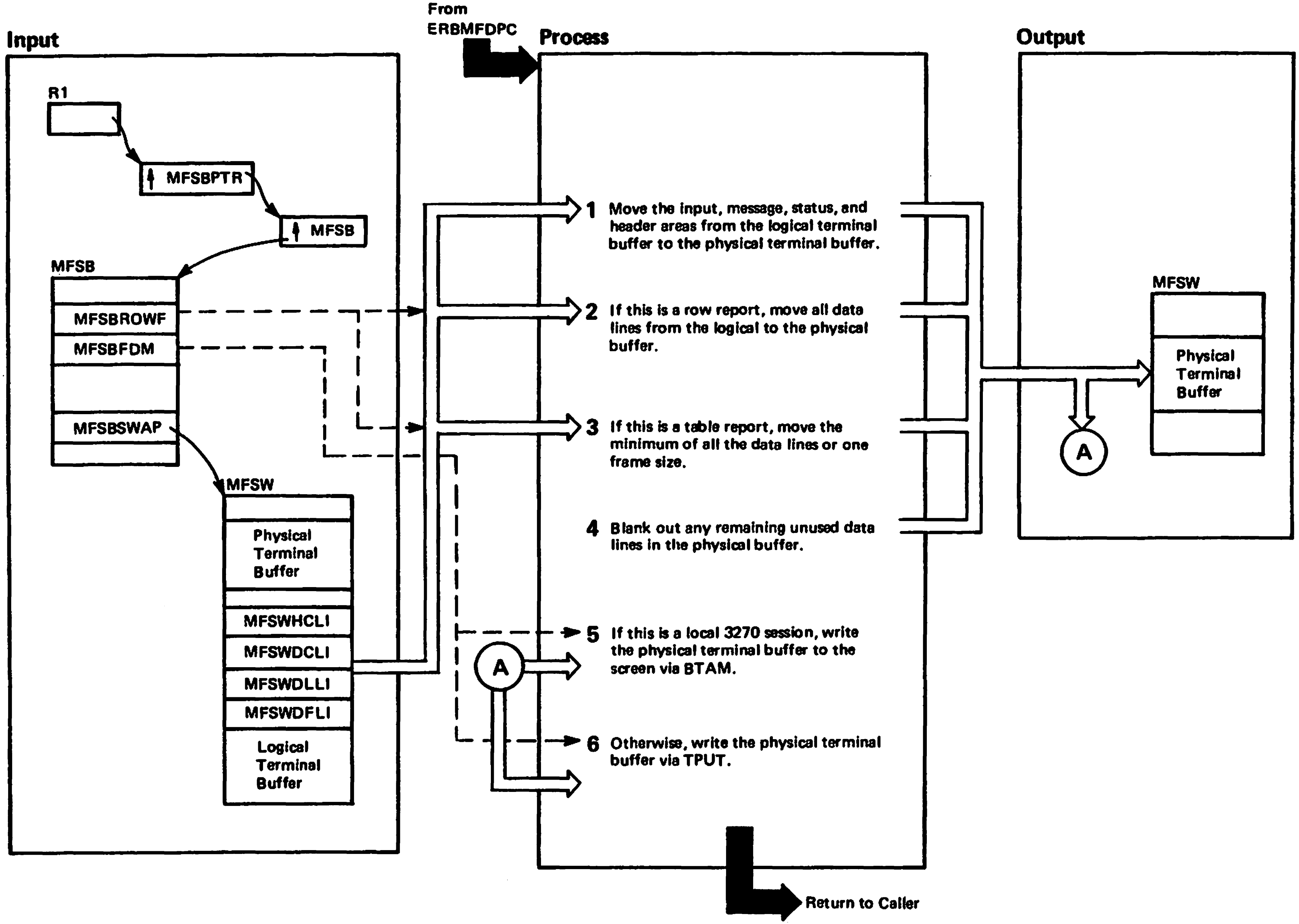


Diagram 90. Terminal Write (ERBTERMW) (Part 2 of 2)

Extended Description	Module	Subroutine

ERBTERMW moves input, message, status, header, and data areas from the logical to the physical terminal buffer, blanks out the remaining unused portion of the physical buffer, and then writes out the buffer to the terminal via BTAM or TPUT.

1

2 If this is a row report (MFSBROWF is on), then there will be at most one frame of data. Move all data lines from the logical to the physical buffer.

3 If this is a table report, then there may be more than one frame of data. Starting with the line pointed to by the frame index (MFSWDFLI), move either one frame size or the data that is left, whichever is smaller.

4

5 If this is a local 3270 session (MFSBFDM bit in MFSBENVC is on), then write the physical buffer to the terminal via BTAM, using the WRITE TI macro instruction. Issue the WAIT macro instruction to wait for the I/O to complete. If either the return code from the WRITE is non-zero or the post-code from the WAIT is '41'X, then write an error message to the operator's console, turn on the error-terminate bit in the MFSB, and return.

6 Otherwise, if this is a TSO session, write the physical buffer to the terminal via a fullscreen TPUT. If the return code is X'10', issue an error message with a normal TPUT, turn on the error-terminate bit, and return.

Diagram 91. Transaction Activity Parsing Routine (ERBTRXS1) (Part 1 of 2)

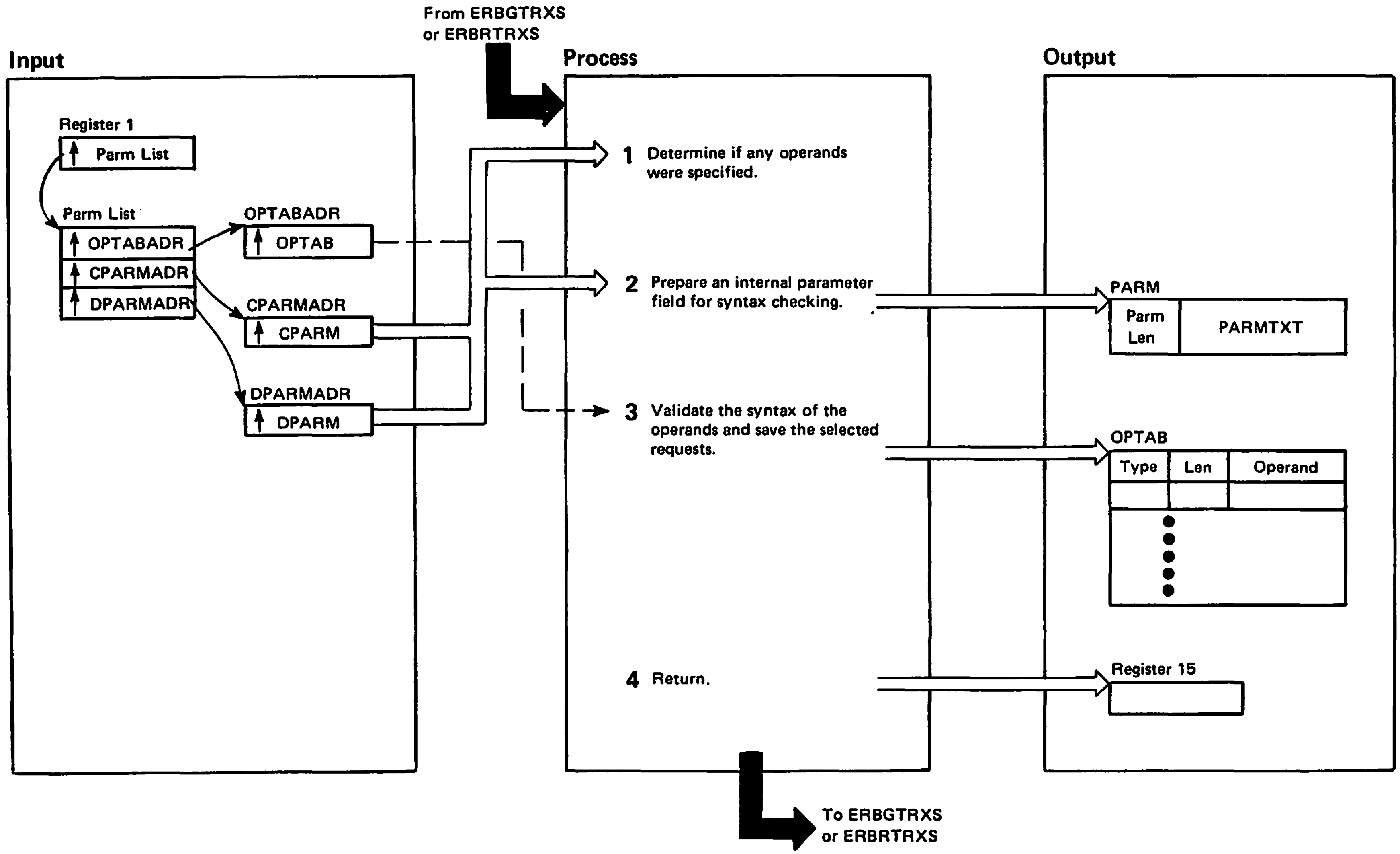


Diagram 91. Transaction Activity Parsing Routine (ERBTRXS1) (Part 2 of 2)

Extended Description	Module	Label
The parsing routine receives control from either the Transaction Activity Data Gatherer (ERBGTRXS) or the Transaction Activity Data Reporter (ERBRTRXS). In either case, three addresses are passed as parameters to this routine, providing access to the following data areas:		
OPTAB — An array used after the syntax checking is done to store the operands, their lengths, and a code describing the type of each operand		
CPARM — The input operand string text and length fields		
DPARM — The menu default operand string text and length fields		
1 If both the CPARM and DPARM length fields contain zero, then the user did not specify any operands. Use the default value ALLPGN as the operand text. If the CPARM length field does not equal zero, use the value(s) in the CPARM text. Otherwise, use the values in the DPARM text.	ERBTRXS1	ERBTRXS1
2 Move the appropriate operands, as described in step 1, into the internal parameter field PARM. Translate the text to numeric codes for syntax checking.	ERBTRXS1	ERBTRXS1
3 Determine if the syntax of the text is valid by scanning PARMTXT and isolating one operand at a time. Each valid operand is placed into a separate OPTAB entry in character form (EBCDIC) along with its length and one of the following type codes (binary) 1 — Subsystem name 2 — Single performance group number 3 — Range of performance group numbers 4 — ALLPGN	ERBTRXS1	ERBTRXS1
4 If a syntax error is detected, terminate processing and return to the calling routine. ERBTRXS1 return codes: 0 — Processing is successful 4 — CPARM syntax error 24 — DPARM syntax error		

ERBPARSE - MODULE DESCRIPTION

DESCRIPTIVE NAME: I/O Queuing/Device Activity Parsing for MonitorII

FUNCTION:
Parses the operands of the IOQUEUE/DEV/DEVV menu items (for Monitor II display sessions) or the suboptions of the IOQUEUE/DEV/DEVV options (for Monitor II background sessions).

ENTRY POINT: ERBPARSE

PURPOSE: See FUNCTION.

LINKAGE:
```
   BALR from ERBGIOQ or ERBGIGQ
          or ERBGDEV or ERBRDEV
```

CALLERS:
```
   ERBGIOQ  - I/O queuing data gatherer (308X/4381),
              Monitor II
   ERBGIGQ  - I/O queuing data gatherer (IBM 3090),
              Monitor II
   ERBGDEV  - Device data gatherer,
              Monitor II
   ERBRDEV  - Device data reporter,
              Monitor II
```

INPUT:
```
   Input parameters are as follows:
   1. CPARM     - Command parm - length/text of input.
                  Contains suboptions of IOQUEUE/DEV/DEVV
                  option (if background session) or
                  operands of the IOQUEUE/DEV/DEVV
                  menu item (if display session).
   2. DPARM     - Default parm - length/text of input.
                  Contains defaults specified in entries
                  of the background menu table ERBBMENU
                  (if background session) or in entries of
                  the foreground menu table ERBFMENU
                  (if display session).
   3. PRSFLG    - Parsing flags. Indicates the menu item
                  or options (DEV/DEVV/IOQUEUE) for which
                  the operands or suboptions should be
                  parsed.
   4. CLASSX    - Class index (output field)
   5. OPENTRY   - Sublist (output field)
```

OUTPUT:
```
   PRSFLG    - Indicates whether the source of the operand
               is a default.
             - Type of keyword (compound or simple)
               that was parsed.

   If a simple keyword:
   CLASSX    - represents the index of the device class.
               1 = TAPE
               2 = COMM
               3 = DASD
               4 = GRAPH
               5 = UNITR
               6 = CHRDR
               7 = NUMBER (if compound keyword)

   If a compound keyword:
   OPENTRY - Contains the sublist of device numbers or
             LCU ids or volume serial numbers
```

EXIT NORMAL: Return to caller.

ERBPARSE - MODULE DESCRIPTION (Continued)

EXTERNAL REFERENCES: See below

ROUTINES: None

CONTROL BLOCKS: None

ERBPARSE - MODULE OPERATION

1. Initializes fields and sets the program default for device class to DASD.

2. Validates the command parameter (CPARM), or, if CPARM is zero, the default parameter (DPARM).

   A. If a simple keyword (see notes) was requested, parses it. If the keyword is valid, sets the device class index. If the keyword is invalid, sets an error return code.

   B. If a compound keyword (see notes) was requested, parses it. If the keyword is valid, builds a sublist that contains the subentries specified for the compound keyword. If the keyword is invalid, sets an error return code.

3. Returns to caller.

Note:

The command parameter (CPARM), or the default (DPARM) parameter, consists of one and only one of the following list of keywords:

1. Simple keywords such as: TAPE, COMM, DASD, GRAPH, UNITR, CHRDR
2. The compound keyword:
   NUMBER=(YYY:YYY,YYY,....,YYY)
   The numbers may be from one to three alphameric characters.
3. For DEV/DEVV menu item/options only, compound keywords such as:
   VOLSER=(XXXXXX,XXXXXX,,XXXXXX)
   The volume serial numbers must be six alphameric characters.

   A compound keyword can be truncated continuously to a minimum of one character; for a example:
   NUMBER,.....,NUMB,...,N
   is acceptable.

   Only one keyword can be specified. All leading commas, intervening commas, and trailing commas are ignored. The maximum number of subentries allowed is sixteen.

RECOVERY OPERATION:

This module is protected by the RMF Monitor II session recovery routine ERBESTAE.

ERBPARSE - DIAGNOSTIC AIDS

ENTRY POINT NAME: ERBPARSE

MESSAGES: None

ABEND CODES: None

WAIT STATE CODES: None

RETURN CODES:

EXIT NORMAL:

```
0 -  Successful completion.
4 -  Bad command input.
24 - Bad default parameter.
```

REGISTER CONTENTS ON ENTRY:

```
Register  1 - Address of parameter list of 5 pointers.
Registers 2 -12  Irrelevant
Register 13 - Address of save area.
Register 14 - Return address.
Register 15 - Entry address.
```

REGISTER CONTENTS ON EXIT:

EXIT NORMAL:

```
Register  0 - Register 14  Restored to their original
              values
Register 15 - Return code
```

ERBGIOQ - I/O queuing data gatherer (308X/4381), Monitor II ERBGIGQ - I/O queuing data gatherer (IBM 3090), Monitor II ERBGDEV - Device data gatherer, Monitor II ERBRDEV - Device data reporter, Monitor II

PARAMETERS

CPARM	DPARM
PRSFLG	CLASSX
OPENTRY	

ERBPARSE

Parses the operands of the IOQUEUE/DEV/DEVV menu items (for Monitor II display sessions) or the suboptions of the IOQUEUE/DEV/DEVV options (for Monitor II background sessions).

01 Initializes fields and sets program default for device class index to DASD.

PARAMETERS: CLASSX

02 If no operand/suboption was provided, uses the program default value (DASD).

03 If a simple keyword was requested, parses the keyword without a sublist.

A. If the length exeeds the maximimum allowed, sets an error return code.

PARAMETERS: CLASSX

B. Otherwise, scans for device class keyword.

PARAMETERS: CLASSX

PARAMETERS: CLASSX

C. If an invalid keyword is encountered, sets an error return code.

D. Otherwise, CLASSX contains the device class.

E. Sets the simple keyword entry indicator in the parsing flags.

04 If a compound keyword was requested, parses the keyword with sublist entries.

A. Sets the compound keyword indicator in the parsing flags.

B. If the keyword is missing or if it exceeds the maximum length, sets an error return code.

C. If the specified keyword is invalid or VOLSER was specified for IOQUEUE, sets an error return code.

D. Otherwise, parses the subentries and moves them into a VOLSER/number list. If an invalid subentry is detected, sets an error return code.

E. If no subentry is found or if the second half of a range is missing, sets an error return code.

05 If an error return code was set, the code indicates the source of the operand (bad operand or bad default operand).

06 Returns to the caller.

Diagram 93. PUTLINE (ERBRMFPL) (Part 1 of 2)

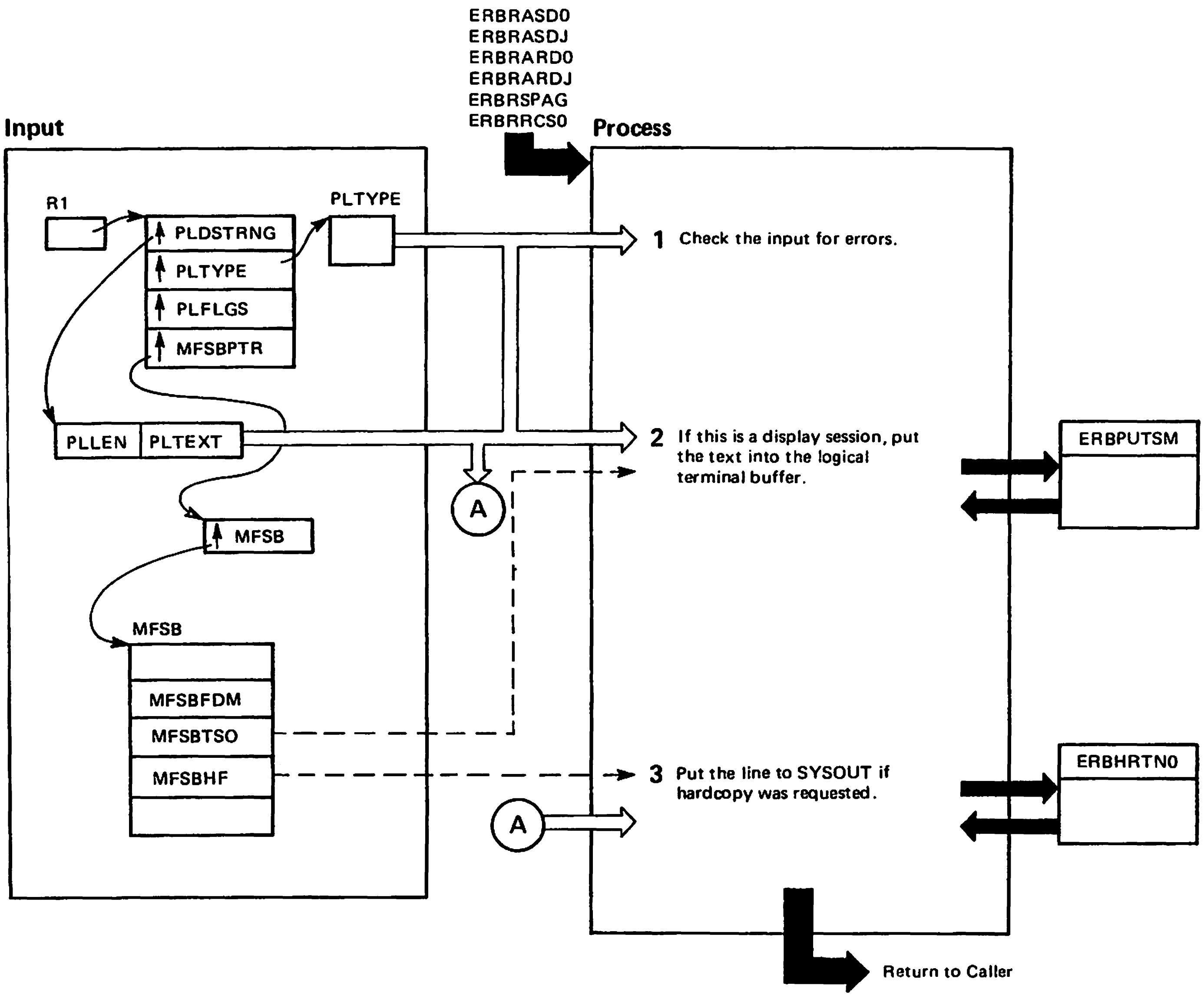


Diagram 93. PUTLINE (ERBRMFPL) (Part 2 of 2)

Extended Description	Module	Subroutine
ERBRMFPL provides the output interface for the data reporters. It calls the HARDCOPY and PUTSTREAM routines if required.		

1 Check the following input:

a. Input text must not be 0 or greater than 79 characters in length.

b. The type of input text must be either header ('HD') or data ('DT'). This check ensures that an unauthorized program, namely, a data reporter, cannot update control information in the terminal buffer.

If either of these checks fails, abend with a user completion code of 1403.

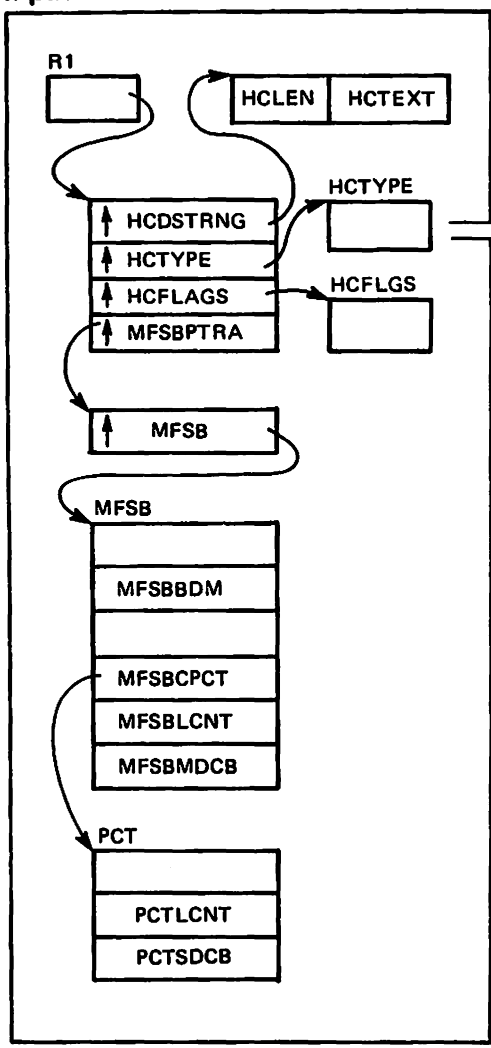
2 If this is a local 3270 or TSO session (MFSBFDM or MFSBTSO bit in MFSB is on), then call ERBPUTSM to put the text in the logical terminal buffer in the screen workarea (MFSW).

3 If the MFSBHF bit in the session control block (MFSB) is on, indicating that HARDCOPY has been requested, call ERBHRTN0 to put the text to SYSOUT.

If Reg 15 ¬= 0, indicating that a synad error occurred, return with the same return code. Otherwise return with a return code of 0.

Diagram 94. ERBHRTN0 Subroutine (ERBRMFPL) (Part 1 of 2)

Input



ERBRMFPL
ERBRESET
ERBMFDPC

Process

1 If this is a background message, put the message to the session data set.

2 If the text type is 'RS', then skip to a new page if necessary, and go to step 4.

3 Save the header line if this is a background session.

4 If a new page is needed, print the headers at the top of the new page.

5 Move the text string to an output buffer, and put the buffer to SYSOUT.

Return to Caller

Output

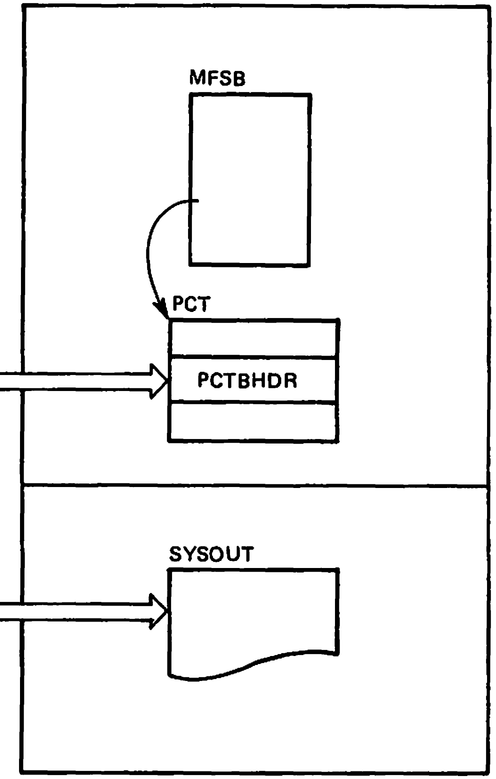


Diagram 94. ERBHRTN0 Subroutine (ERBRMFPL) (Part 2 of 2)

Extended Description | **Module** | **Subroutine**

ERBHRTNO puts a line of text to SYSOUT.

1 If the text is a message for a background session, put the text to the session SYSOUT data set (its DCB address is in the MFSB (MFSBMDCB)). If the text should not also be put to the report SYSOUT data set (its DCB address is in the PCT (PCTSDCB)), then return.

2 If text type is 'RS', then text string is a delimiter line. Skip to a new page if the delimiters, headers, and the first data line won't fit on the old page. Go to number 4.

3 If this is a background session (MFSBBDM is on) and a header line (HCTYPE='HD'), then save the line in the PCT (PCTBHDR).

4 If a new page is needed, print the headers at the top of the new page.

5 Use the PUT macro instruction to print the line.

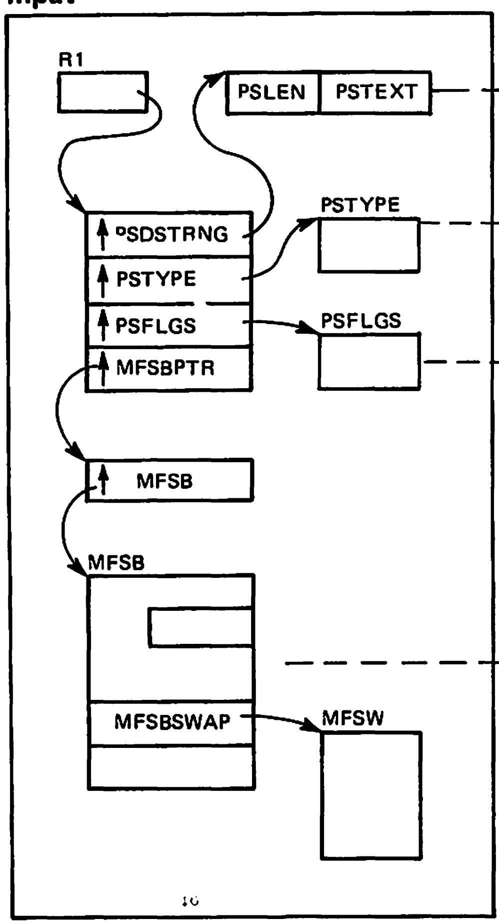
After each PUT, check whether a synad error occurred. If it did (MFSBSYN is on), and this is a display session, then turn off the hardcopy flag (MFSBMF). Return to caller since further processing would produce more errors. The return code is dependent upon the text, the data set, and the session:

RC = 4 display session, report or delimiter lines for background session.

RC = 8 background message going to report SYSOUT data set.

RC = 12 background message going to session SYSOUT data set.

Diagram 95. ERBPUTSM Subroutine (ERBRMFPL) (Part 1 of 2)



ERBRMFPL
ERBMFDPC

Process

1 If text type is input, message, migration age, status, or header, move the text into the appropriate area in the logical terminal buffer.

2 If the text type is data:

a. If this is a row report:

1. Update the highlighting flags.
2. Update the line indices.
3. Move the text string into the line.

b. If this is a table report:

1. Update the line indices.
2. Move the text string into the line.

Return to Caller

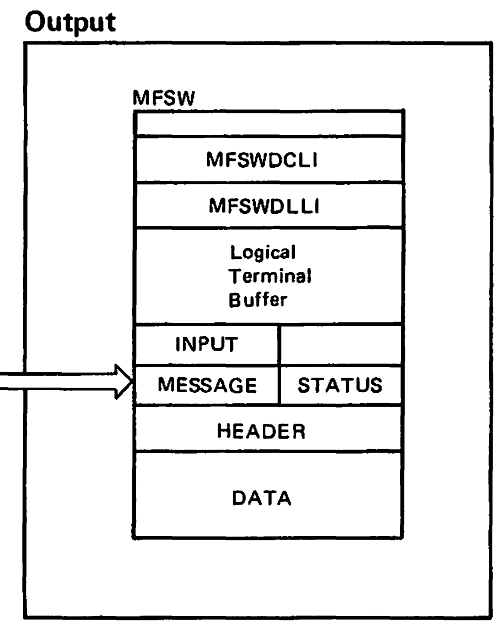


Diagram 95. ERBPUTSM Subroutine (ERBRMFPL) (Part 2 of 2)

Extended Description	Module	Subroutine

ERBPUTSM inserts a text string into the logical terminal buffer.

1 If PSTYPE is input ('IN'), message ('MG'), migration age ('MI'), status ('ST'), or header ('HD'), check that the length of the text string is not 0. If it is, abend with a user completion code of 1405. Otherwise, move the text string into the appropriate area in the logical terminal buffer, and highlight the field if the highlighting flag is on. For header type, first check that another header line will fit into the logical terminal buffer. If it won't, abend with a user completion code of 1404.

2 If PSTYPE is data ('DT'):

a. If this is a row report:

(1) If input text is not the first line of data, then change the attribute flags of the previous line to no highlighting.

(2) Increase the data last line index (MFSWDLLI) by one if it is less than 21 (number of data lines in frame). Increase the data current line index (MFSWDCLI) by one, module 21.

(3) Move the text string into the line.

Set the attribute flags to highlight.

b. If this is a table report:

(1) Check that another data line will fit into the logical terminal buffer. If it won't, abend with a user completion code of 1404. Increment data current and last line indices by one.

(2) Move the text string into the line, and highlight the field if the highlighting flag is on.

Diagram 96. ERBRESET Subroutine (ERBRMFPL) (Part 1 of 2)

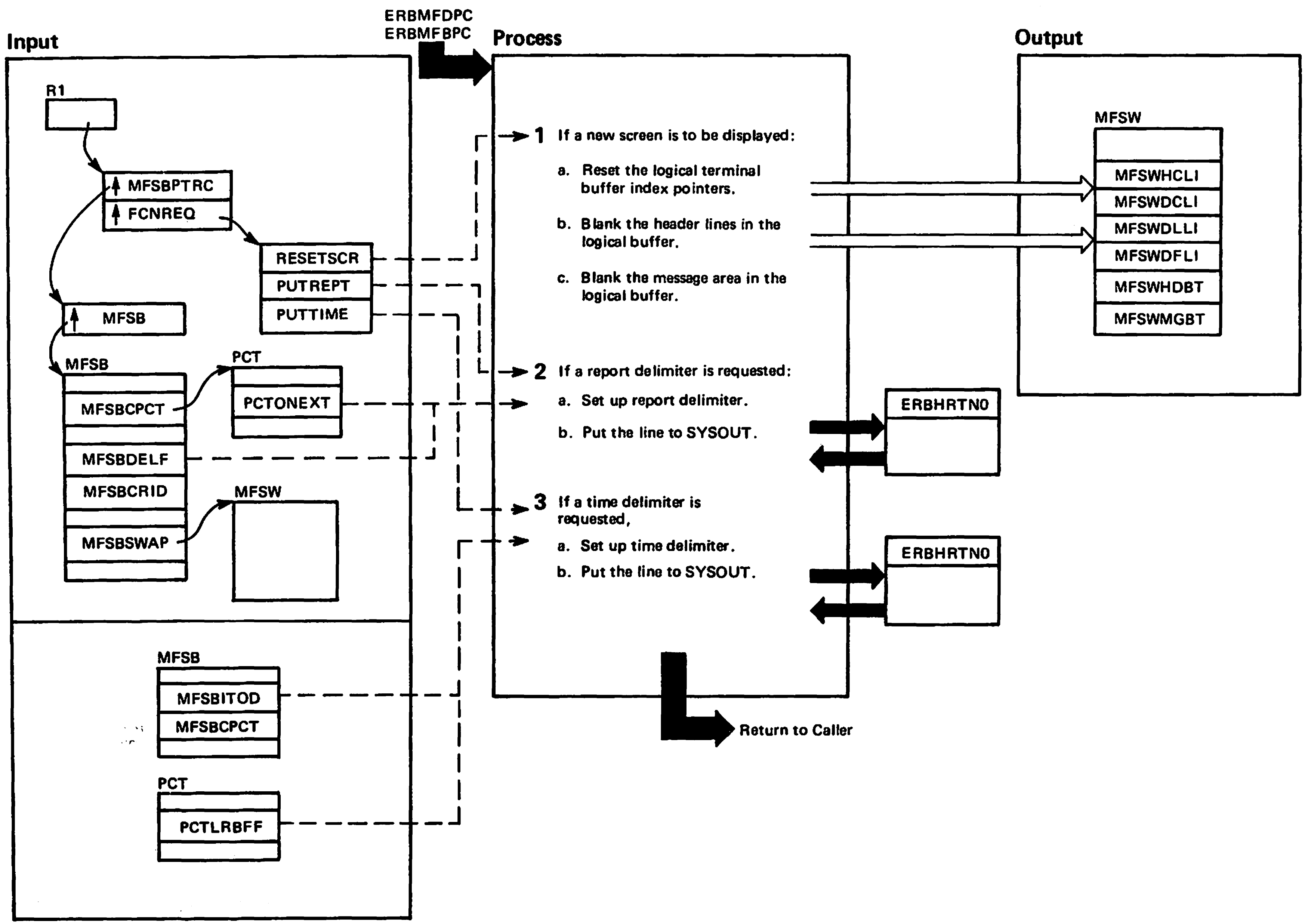


Diagram 96. ERBRESET Subroutine (ERBRMFPL) (Part 2 of 2)

Extended Description	Module	Subroutine
ERBRESET prepares for a new report.		
1 If the RESETSCR flag is on in the parameter function byte (FCNREQ):		
a. Reset the logical buffer index pointers (data and header current index, data last index, and frame index).		
b. Blank out the header lines in the logical terminal buffer.		
c. Blank the message area in the logical buffer.		
2 If the PUTREPT flag is on in FCNREQ:		
a. Set up the report delimiter to show the current operands, DELTA or TOTAL mode, and the report name.		
b. Call ERBHRTN0 to put the line to SYSOUT.		
3 If the PUTTIME flag is on in FCNREQ:		
a. Set up the time delimiter to show the interval time-of-day, and the status information, if available.		
b. Call ERBHRTN0 to put the line to SYSOUT.		
If ERBHRTN0 returns a code of 4, indicating that a synad error occurred, return with R15=4. Otherwise, return with R15=0.		

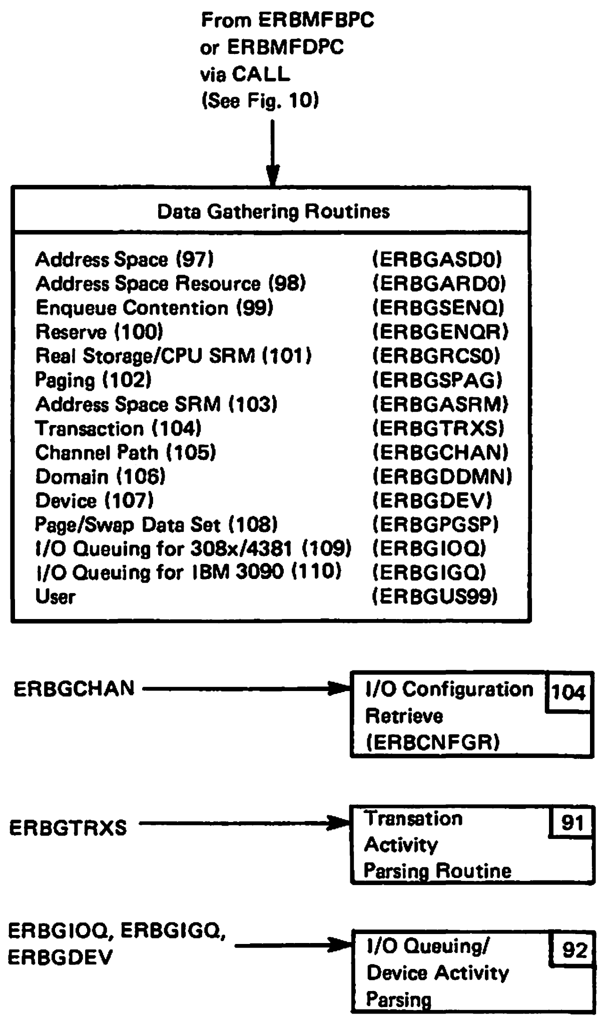


Figure 11. Monitor II Data Gathering Overview

Diagram 97. Address Space State Data Gatherer (ERBGASD0) (Part 1 of 10)

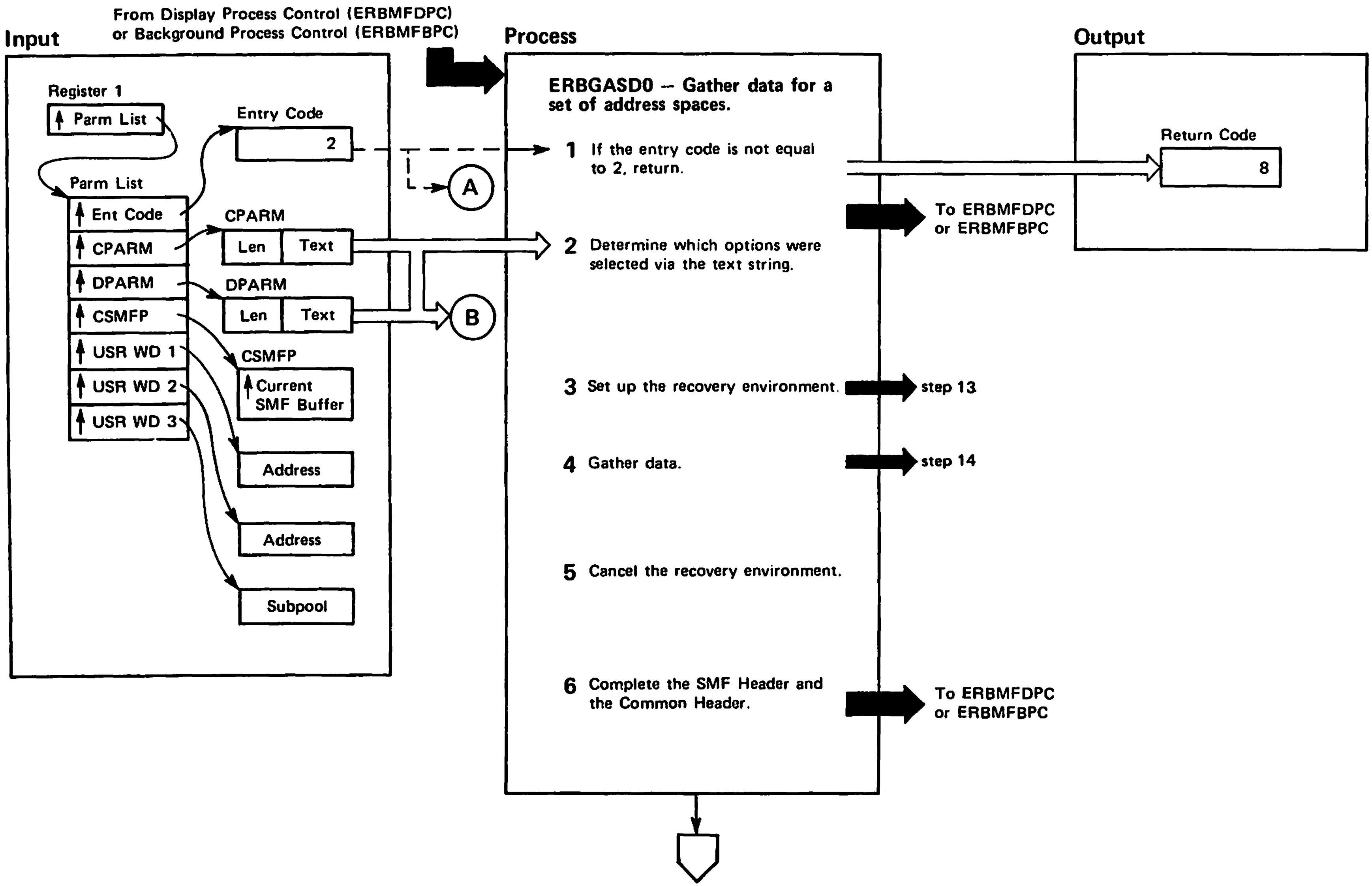


Diagram 97. Address Space State Data Gatherer (ERBGASD0) (Part 2 of 10)

Extended Description	Module	Label

The Address Space Data Gatherer collects data for a specified set of address spaces and builds an internal image of an SMF record.

ERBGASD0 Entry – Gather data for a set of address spaces

1 Check the entry code. If it is not equal to 2, which means go ahead and process, return with a code of 8. — ERBGASD0 — ERBGASD0

2 Do the following to determine which options for CLASS, STATUS, and DOMAIN are to be used for selecting the address spaces reported. — ERBGASD0 — PARSE

Initialize the internal flags to zero. Parse the CPARM text which comes from the command. Set option flags as new options are found. If there are any syntax errors in CPARM, return with a code of 4.

If any of the 3 options have not been chosen, parse the DPARM text which comes from the menu (IBM-supplied or user-modified). Set any option flags which were not set previously from CPARM. If there are any syntax errors in DPARM, return with a code of 24.

If any options are still to be determined, default them to ALL.

The following table shows keyword flag settings:

Keyword	*Type*	*Flag Settings*	*Meaning*
A	class	BCL=ON, TSOCL=ON	All classes
B	class	BCL=ON, TSOCL=off	Batch, Started Task, Mount Tasks
T	class	bCL=off, TSOCL=on	TSO
A	status	ALLST=on	All states – In, Out Ready, Out Wait
I	status	ALLST=off	Swapped In or Swapped Out but eligible for swap in (Non waiting tasks)
A	domain	ALLDMN=on	All domains
nnn	domain	ALLDMN=off, DMNUMB=binary value of nnn	Specific SRM domain

Extended Description	Module	Label

3 Call an internal subroutine to set up the ESTAE environment (step 13). — ERBGASD0 — SETUP

4 Call an internal subroutine to scan the Address Space Vector Table (ASVT) and gather data (step 14). — ERBGASD0 — SCAN

5 Cancel the Estae Exit by issuing the ESTAE macro with an exit of zero. — ERBGASD0 — ERBGASD0

6 Fill in the remaining SMF record fields in the Common Section:

SMF79STY = 1, record subtype for ASD and ASDJ pictures

SMF79ASN, the final count of the number of relocate blocks (1 per address space)

- Set the number of data control sections to zero (SMF79DCN=0)
- Set the length of data control section to zero (SMF79DCL=0)
- Set the offset to the data section SMF79ASS = Address of header of SMF79 record + SMF79DCS + (SMF79DCL *SMF79DCN)
- Set the length of the data section SMF79ASL = length (R791ELEM)

All other fields which have not been filled in by ERBGASD0 were set by Picture Build (ERBPCTBL), and if background, by the ERBBDATA subroutine of Background Process Control (ERBMFBPC).

Return to caller.

Diagram 97. Address Space State Data Gatherer (ERBGASD0) (Part 3 of 10)

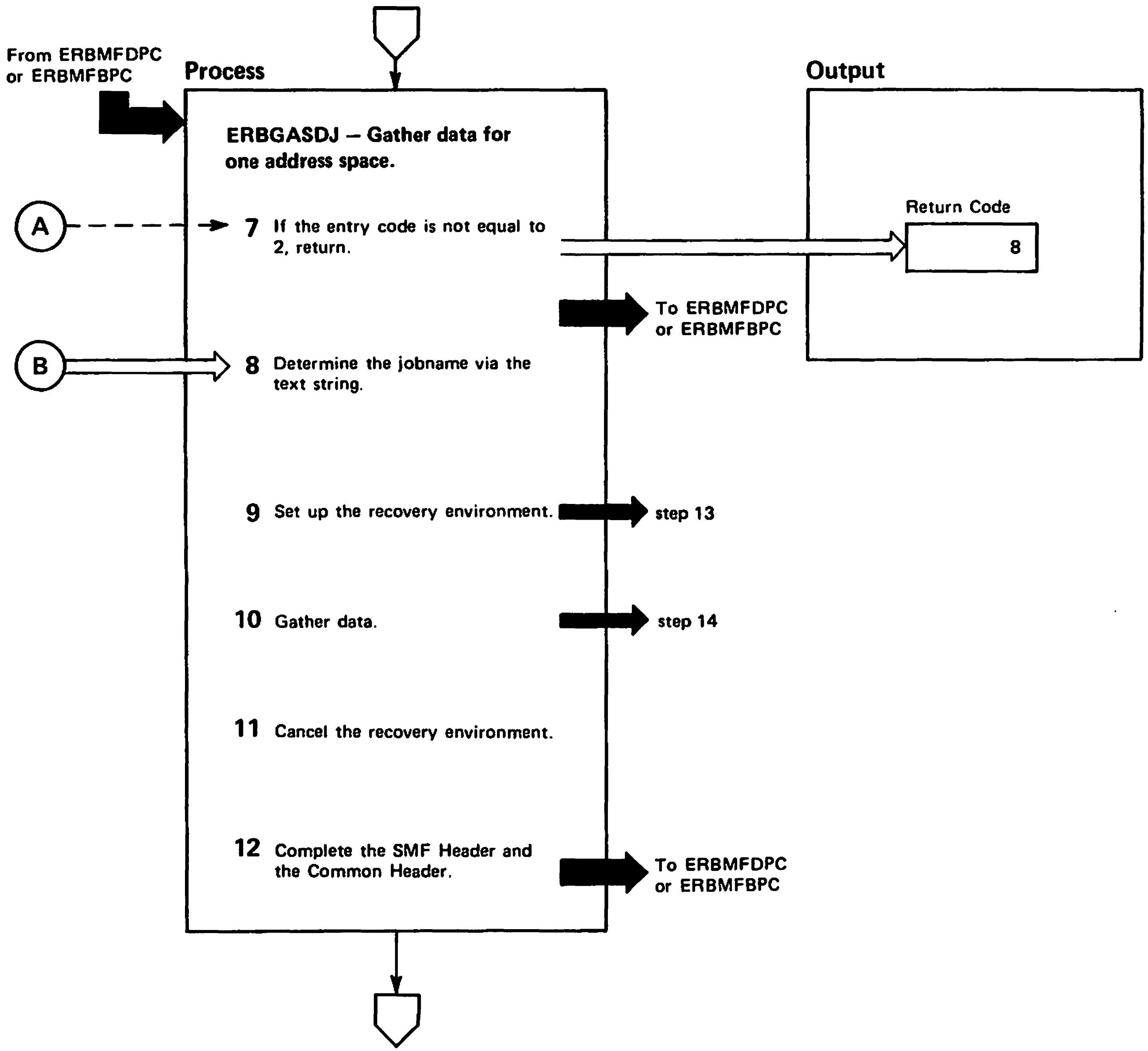


Diagram 97. Address Space State Data Gatherer (ERBGASD0) (Part 4 of 10)

Extended Description	Module	Label
ERBGASDJ Entry — Gather data for one address space		
7 Check the entry code. If it is not equal to 2, which means go ahead and process, return with a code of 8.	ERBGASD0	ERBGASDJ
8 Determine which jobname was requested. Parse the CPARM text. If it is null (length = 0), parse the DPARM text. Set flag (JBN=ON) to indicate search for one job. If there is an error in the text string (greater than 9 characters), return with a code of 4.	ERBGASD0	ERBGASDJ
9 Call an internal subroutine to set up the ESTAE environment (step 13).	ERBGASD0	SETUP
10 Call an internal subroutine to scan the ASVT and gather data (step 14).	ERBGASD0	SCAN
11 Cancel the Estae Exit by issuing the ESTAE macro with an exit address of zero.	ERBGASD0	ERBGASDJ

12 Fill in the remaining SMF record fields in the Common Section:

SMF79STY = 1, record subtype for ASD + ASDJ pictures

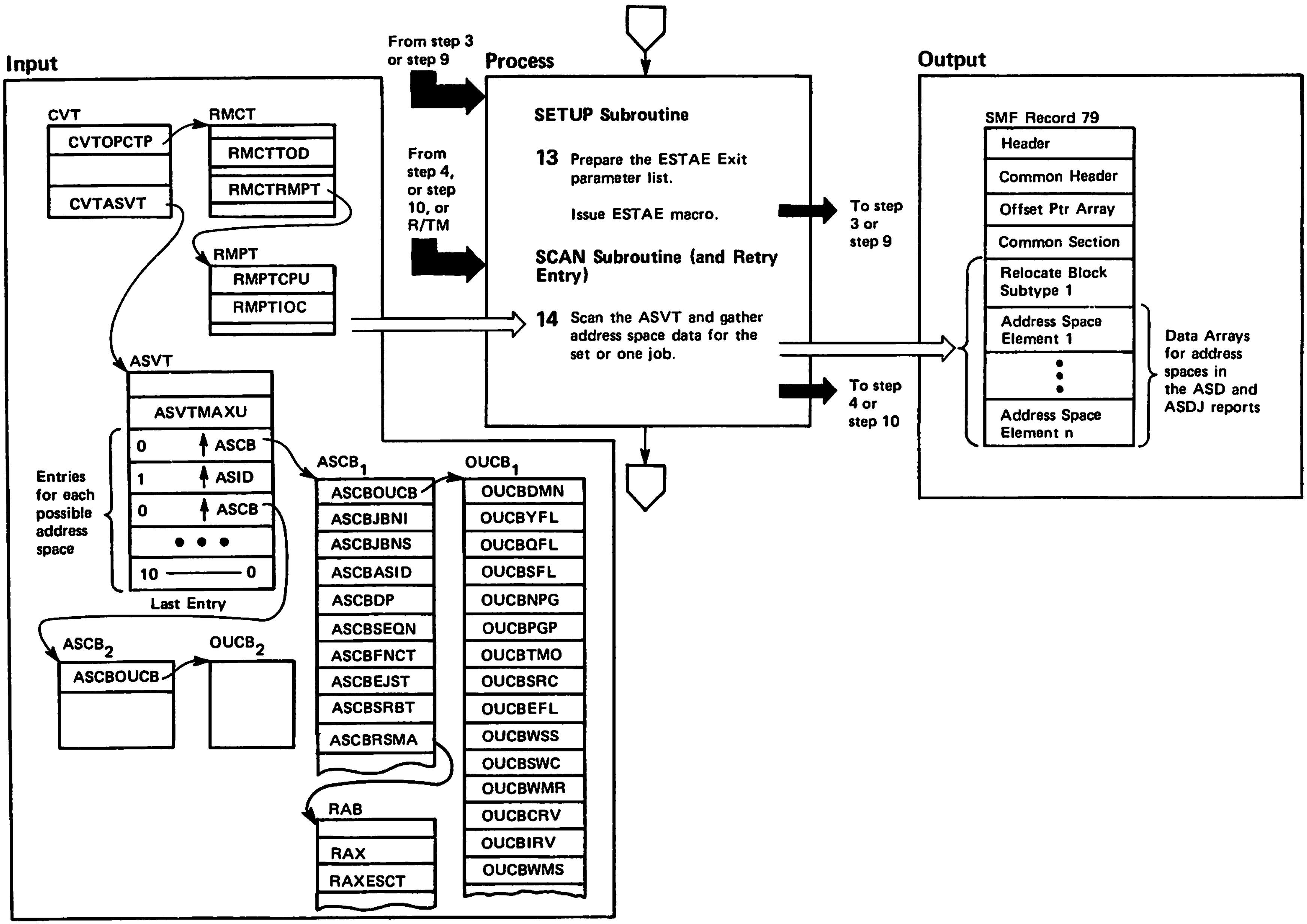
SMF79ASN, the final count of the number of relocate blocks (1 per address space)

- Set the number of data control sections to zero (SMF79DCN = 0)
- Set the length of data control section to zero (SMF79DCL = 0)
- Set the offset to the data section (SMF79ASS) SMF79ASS = Address of SMF79 record + SMF79DCS + (SMF79DCL *SMF79DCN)
- Set the length of the data section SMF79ASL = length (R791ELEM)

All other fields which have not been filled in by ERBGASD0 were set by Picture Build (ERBPCTBL), and if background, by the ERBBDATA subroutine of Background Process Control (ERBMFBPC).

Return to caller.

Diagram 97. Address Space State Data Gatherer (ERBGASD0) (Part 5 of 10)



Extended Description	Module	Label
SETUP Subroutine		
13 Prepare a parameter list with:	ERBGASD0	SETUP
• Counter of the number of retries		
• Save area for registers 0 through 15		
• Address of retry entry point.		
Initialize the counter of retries to 0 and the retry address to the address of RETYRPT.		
Issue ESTAE macro to set up ESTAE Exit (ESTAEASD).		
The Estae Exit will cover any errors in processing the Address Space Vector Table (ASVT) chain of Address Space Control Blocks (ASCB's).		
Return to step 3 or 9.		
SCAN Subroutine (Retry Entry — RETRYPT)		
14 If the number of retries is zero, it is the first entry to this subroutine. Therefore, store the registers in the parameter list save area.	ERBGASD0	SCAN
If the number of retries is not zero, it is not the initial entry. An abend has occurred, and the retry was successful. Therefore, load the registers from the parameter list save area and increase the count of the number of retries by one.		
Gather the address space data that is required by the user:	ERBGASD0	SCAN
a. Get the current time of day with the TIME macro in decimal (HHMMSSTH). Save in R79GTOD in the form 0HHMMSSF (hours, minutes, seconds, sign).	ERBGASD0	SCAN
b. Initialize the number of relocate block elements (number of R791ELEMs) to 0 (SMF79ASN = 0). Set the size of each element (SMF79ASL = length (R791ELEM)).		
c. If a single address space was requested by jobname (JBN=ON), then scan the Address Space Vector Table (ASVT) for an Address Space Control Block (ASCB) whose ASCBJBNI or ASCBJBNS field points to a jobname which is equal to the requested jobname.	ERBGASD0	SCAN
If it is found, gather the data for the address space from its ASCB and Resources Manager Control Block (OUCB) — step 14e.		
If it is not found, return with a code of 16.		
d. If a set of address spaces is requested (JBN=OFF), then search the ASVT for the ASCB's that meet the CLASS/STATUS/DOMAIN criteria specified. (Check the SMF record internal flags set after parse in step 2.)	ERBGASD0	SCAN
• DOMAIN If ALLDMN=ON, then any domain was requested, so check the ASCB further for CLASS. Or if ALLDMN=OFF and OUCBDMN=DMNUMB, then a specific domain was requested and this ASCB is in that domain, so check the ASCB further for CLASS. Otherwise, skip this ASCB and try the next one.		
• CLASS If TSOCL=ON and OUCBLOG=ON, then TSO tasks were requested and this ASCB is TSO, so check it further for STATUS. Or if BCL=ON and OUCBLOG=OFF, then non-TSO tasks were requested, and this ASCB is non-TSO, so check it further for STATUS. Otherwise, skip this ASCB and try the next one.		
• STATUS If ALLST=ON then all tasks (in, out ready and out wait) were requested, so gather data for this ASCB since it meets all requirements. Or if ALLST=OFF and OUCBOUT=OFF, then only in or non-waiting tasks were requested and this ASCB is "in", so gather data for it. Or if ALLST=OFF and OUCBOUT=ON and OUCBOFF=OFF, then only in or non-waiting tasks were requested and this ASCB is out but not waiting, so gather data for it. Otherwise, skip this ASCB and try the next one.		
If no address spaces meet the criteria, return with a code of 16.		

Diagram 97. Address Space State Data Gatherer (ERBGASD0) (Part 7 of 10)

No diagram.

Extended Description continued on next page.

Extended Description	Module	Label

14 (continued)

e. Gather the following information for each address space that meets the requirements. Save the data in the current SMF buffer. — Module: ERBGASD0 — Label: GATHER

- SMF79ASN — increase by 1 to count the number of address spaces reported on
- R791ASID = ASCBASID, address space identifier
- R791JBN = field pointed to by ASCBJBNI or ASCBJBNS, i.e., jobname (initiated programs-batch) or jobname (start/mount/logon)
- R791DMN = OUCBDMN, domain number
- R791NPG = OUCBNPG, new performance group number
- R791PGP = ((OUCBPGP – (length (WPGD) – length (WPGP))) ÷ length (WPGP)) + 1, performance period
- R791CL=x, current location
  where x =
  - 'IN' — assume (swapped in), then check further - may overlay
  - 'PR' — if OUCBPVL = ON, (user program privileged), check further - may overlay
  - 'NS' — if OUCBNSW = ON (non-swappable status), quit checking
  - 'WM' — if OUCBOFF = ON and OUCBMWT = ON (request enter wait state and MSO detected wait status), quit checking
  - 'WT' — if OUCBOFF = ON and OUCBTRM = ON (request enter wait state and terminal status wait), quit checking
  - 'WL' — if OUCBOFF = ON and OUCBLWT = ON (request enter wait state and long wait status), quit checking
  - 'LO' — if OUCBLSW = ON (logically swapped out), quit checking.
  - 'OT' — if OUCBOUT = ON and OUCBDLYB = OFF (out, and READY by elimination), quit checking.
  - 'DL' — if OUCBOUT = ON and OUCBDYLB = ON (delayed user), quit checking.
  - ' >> ' — if OUCBOUT = OFF and OUCBGOO = ON (Transitioning out), quit checking.
  - ' << ' — if OUCBOUT = OFF and OUCBGOI = ON (Transitioning in), quit checking.
- R791TAS = x, type of user
  where x =
  0 – if ASCBJBNI not 0 (batch)
  1 – if OUCBSTT = ON (started task)
  2 – if OUCBMNT = ON (mount created user)
  3 – if OUCBLOG = ON (TSO)
- R791SRC = x, reason for last swap out – (valid if address space is swapped out, because reset when swapped in)
  where x =

'TO'	– if OUCBSRC = 1	'RQ '	–	if OUCBSRC = 7
'TI'	– if OUCBSRC = 2	'NQ'	–	if OUCBSRC = 8
'LW'	– if OUCBSRC = 3	'EX'	–	if OUCBSRC = 9
'XS'	– if OUCBSRC = 4	'US'	–	if OUCBSRC = 10
'RS'	– if OUCBSRC = 5	'TS'	–	if OUCBSRC = 11
'DW'	– if OUCBSRC = 6	'00'	–	if OUCBSRC = > 11 or < 0

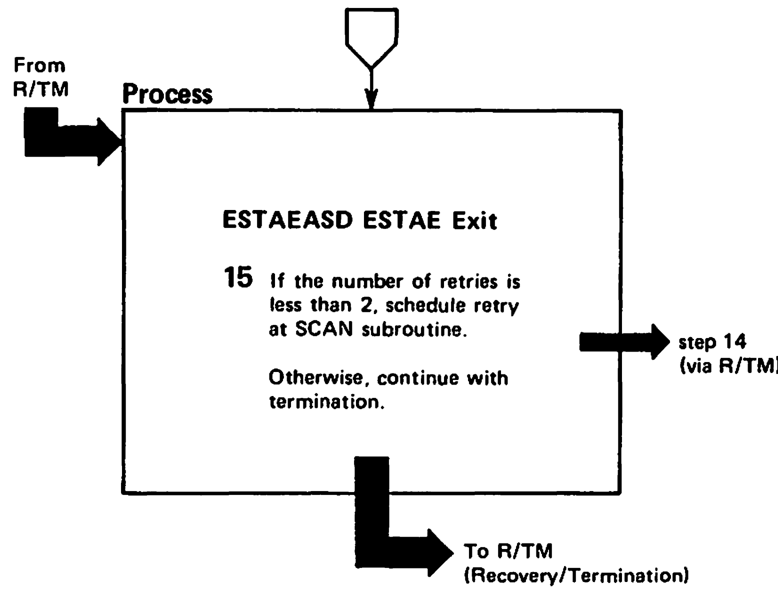
- R791DP = ASCBDP, dispatching priority (range 0-255)
- R791SEQN = ASCBSEQN, ASCB's position on the dispatching queue
- R791FMCT = ASCBFMCT, allocated page frame count (real storage frames)
- R791WSS = OUCBWSS, working set size at swap-in
- R791SWC = OUCBSWC, transaction swap count
- R791SWMR = OUCBWMR/256, SRM workload manager recommendation value
- R791SCRV = OUCBCRV*RMPTCPU/256, SRM CPU recommendation value
- R791SIOC = OUCBIRV*RMPTIOC/256, SRM I/O Manager recommendation value
- R791SSRV = OUCBSBRV * RMPTMSO/256, SRM storage manager recommendation value
- R791ES = RAXESCT, number of pages on extended storage
- R791WMS = OUCBWMS the current transaction's service accumulator (since the last swap-in.)
- R791TCPU = (ASCBEJST + ASCBSRBT) / ((2**12)* (10**3)), elapsed job set timing plus accumulated SRB time (total CPU, TCB, and SRB since step start in milliseconds)

f. When all the data is collected for an address space, if only the swapped in or eligible for swap in set of address spaces was requested (ALLST = OFF), double check the status of this address space to be sure it is still swapped in or eligible for swap in (OUCBOUT = OFF or OUCBOUT = ON and OUCBOFF = OFF). — Module: ERBGASD0 — Label: GATHER

If it still meets the STATUS option, continue with the next ASCB.

If it no longer meets the STATUS option, eliminate its entry in the SMF Record. Decrease SMF79ASN by 1.

Diagram 97. Address Space State Data Gatherer (ERBGASD0) (Part 9 of 10)



Extended Description	Module	Label
15 Check register 0 to determine if the System Diagnostic Work Area (SDWA) was provided. If register 0 = 12 (decimal), it was not. Therefore, get the address of the parameter list from register 2. If the SDWA was provided, the address of the parameter list is in the 1st word of the SDWA. The parameter list has the counter for the number of retries.	ERBGASD0	ESTAEASD

Check the counter of the number of retries. If it is less than 2, indicate that a retry routine should be given control.

- If the SDWA was available, fill in the SDWA with the retry address and a return code of 4.
- If the SDWA was not available, set register 0 to the retry address and pass back a return code of 4 in register 15.

If the number of retries is already 2, don't retry again, just percolate the abend.

- If the SDWA was available, fill in the SDWA with return code 0 (default anyway)
- If the SDWA was not available, return with a code of 0 in register 15.

ESTAE exit return codes in Register 15:

0 — continue termination
4 — retry at address provided

ERBGASD0 return codes:

0 — Data successfully gathered
4 — Syntax error in CPARM string
8 — Invalid entry code
16 — No address space which meets criteria (jobname or class/domain/status)
20 — Unable to establish ESTAE environment.
24 — Syntax error in DPARM string.

Diagram 98. Address Space Resource Data Gatherer (ERBGARD0) (Part 1 of 8)

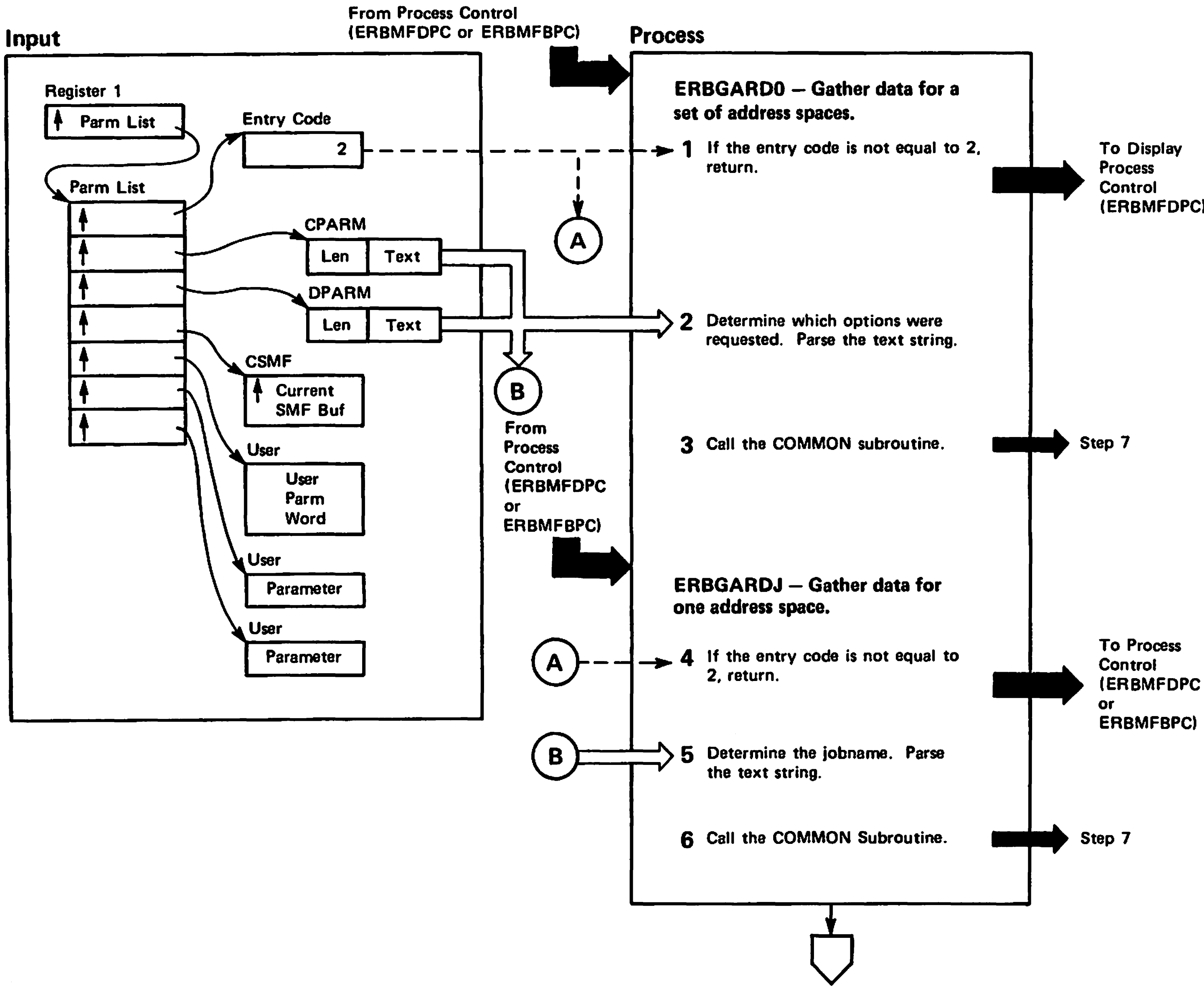


Diagram 98. Address Space Resource Data Gatherer (ERBGARD0) (Part 2 of 8)

Extended Description	Module	Label
The Address Space Resource Utilization Data Gatherer collects data for a specified set of address spaces and builds an internal image of an SMF record.		
ERBGARD0 Entry – Gather Data for a Set (Table Report)		
1 Check the entry code. If it is not equal to 2, return with a code of 8.	ERBGARD0	ERBGARD0
2 Determine which options were requested for CLASS, STATUS, and DOMAIN. Parse the CPARM text.	ERBGARD0	PARSCMD

If any parameters are null, PARSE the DPARM text and use the value from DPARM to default the missing parameter. If the missing parameter is also missing in DPARM, set an internal default. (For defaults see the table below).

Set internal flags based on the options requested. See following table.

Keyword	*Type*	*Flag Settings*	*Meaning*
A	class	BCL=ON, TSOCL=ON	All classes
B	class	BCL=ON, TSOCL=OFF	Batch, Started Task and Mount Task
T	class	BCL=OFF, TSOCL=ON	TSO
I	status	ALLST=OFF	Swapped In or Swapped Out and eligible for swap in
A	status	ALLST=ON	All
nnn	domain	ALLDMN=OFF DMNUMB=binary value of nnn	Specific SRM Domain
A	domain	ALLDMN=ON	All domains

Defaults:

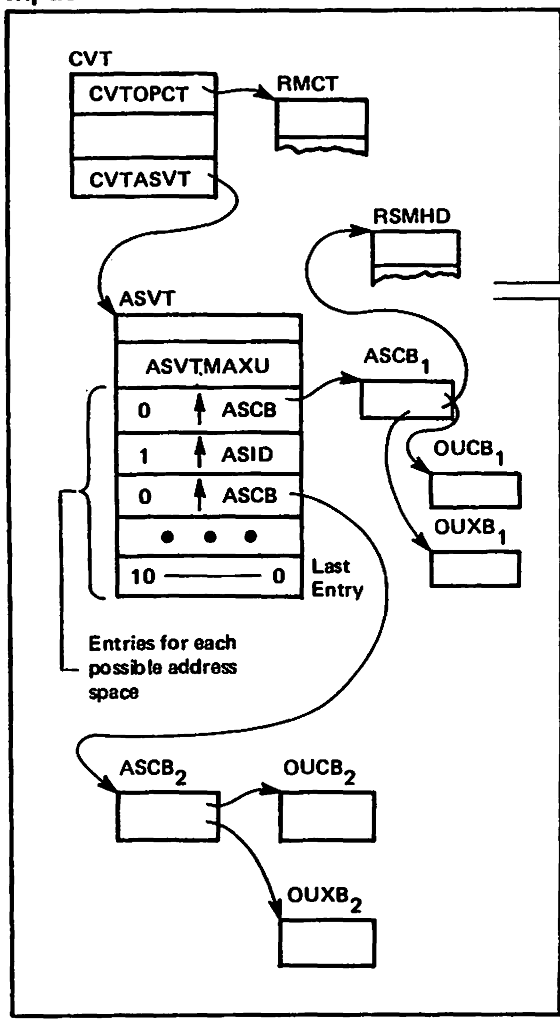
CLASS – All (BCL=ON and TSOCL=ON)
STATUS – All (ALLST=ON)
DOMAIN – All (ALLDMN=ON)

If a syntax error was found in the CPARM text, return with a code of 4.

Extended Description	Module	Label
If a syntax error was found in the DPARM text, return with a code of 24.		
3 Call internal subroutine to set up and gather the data (step 7).	ERBGARD0	ERBGARD0
ERBGARDJ Entry – Gather Data for one job (Row Report)		
4 Check the entry code. If it is not equal to 2, return with a code of 8.		
5 Determine which jobname was requested. Set a flag (JBN=ON) to indicate search for one job.	ERBGARD0	PARSJOBN
If error in text string (greater than 9 characters), return with a code of 4.		
6 Call internal subroutine to set up and gather the data (step 7).	ERBGARD0	ERBGARDJ

Diagram 98. Address Space Resource Data Gatherer (ERBGARD0) (Part 3 of 8)

Input



From step 3 and step 6

ESTAE Retry Entry (RTRYPT)

Process

COMMON Subroutine

7 Set up the recovery environment and prepare for retries.

8 Gather address space data.

C

Extended Description	Module	Label
Common Subroutine		
7 Issue ESTAE macro to set up ESTAE Exit (ABNDEXIT). Initialize the number of retries to 0. Save registers in a Retry Work area for the retry entry point (RTRYPT). The Estae Exit will cover any errors in processing the Address Space Vector Table (ASVT) chain of Address Space Control Blocks (ASCB's).	ERBGARD0	COMMON
8 Gather the address space data that is required by the user:	ERBGARD0	COMMON
a. Get the current time of day and save in R79GTOD. Initialize the number of relocate block elements (number of R792ELEM s) to zero (R792NELM = 0). Set the size of each element in R792SELM.		
b. If a single address space was requested by jobname (JBN=ON), then scan the Address Space Vector Table (ASVT) for an Address Space Control Block (ASCB) whose JOBNAME is equal to the requested jobname. When found, gather the data for the address space — step 8d. If an eligible Address Space was not found, return with a code of 16.	ERBGARD0	CHKELIG
c. If a set of address spaces is requested, search the ASVT for the ASCB s that meet the CLASS/STATUS/DOMAIN criteria specified (check the internal flags set after parse in step 2):	ERBGARD0	CHKELIG
• DOMAIN If ALLDMN=OFF and OUCBDMN≠DMN a specific domain was requested, but this address space is not in that domain. Skip the ASCB, try the next. Else check this address space further for CLASS.		
• CLASS If TSOCL=OFF and OUCBLOG=ON then TSO address spaces were not requested, but this is a TSO address space. Skip the ASCB, try the next. Or if BCL=OFF and OUCBLOG=OFF then only TSO address spaces were requested, but this is not a TSO address space. Skip the ASCB, try the next. Else check this address space further for STATUS.		
• STATUS If ALLST=OFF and OUCBOUT=ON and OUCBOFF=ON, then only swapped in address spaces, or swapped out but eligible for swap in address spaces were requested, but this address space is swapped out and in a wait state. Skip this ASCB, try the next. Else gather data since this ASCB meets all the requirements. If no address spaces meet the criteria, return with a code of 16.		
d. Gather the following information for each address space that meets the requirements. Save the data in the current SMF buffer. • SMF79ASN — add an increment of 1 to count the number of Relocate Blocks • R792ASID = ASCBASID, address space identifier • R792JBN = field pointed to by ASCBJNI or ASCBJBNS, jobname of initiated programs (batch) or jobname of start/mount/logon tasks. • R792DMN = OUCBDMN, domain number • R792NPG = OUCBNPG, new performance group number • R792CL = current location 'IN' — OUCBDOUT = OFF or OUCBOUT = ON and OUCBOFF = OFF 'OU' — OUCBOUT = ON and OUCBOFF = ON 'LO' — if OUCBSLW = ON (logically swapped out), quit checking. • R792TAS = type of user 0 — batch (if ASCBJBNI ≠ 0) 1 — started (if OUCBSTT = ON) 2 — TSO (if OUCBLOG = ON) 3 — mount (if OUCBMNT = ON) • R792TRC = OUCBTRC (only if in core), transaction count • R792TTOD = RMCTTOD — OUCBTMO, real time into transaction (current time of day minus transaction start time) • R792PSS = OUCBPSS, CPU page seconds • R792EJST = (ASCBEJST + ASCBSRBT) / ((2**12)*(10**3)), step total CPU (TCB + SRB) time in milliseconds • R792ARS = R792PSS/R792EJST, step average of real frames • R792TSRM = OUXBJBS + OUXBTRS + OUCBWMS, step total SRM service • R792RTM = OUXBJBR + OUXBTRR + (RMCTTOD — OUCBTMS), step resident time(s) • R792SVAR = R792TSRM/R792RTM, step SRM service absorption rate • R792TCPU = ASCBEJST / ((2**12)*(10**3)), CPU (TCB) time in milliseconds • R792EXCP = ASCBIOSM, step EXCP count	ERBGARD0	CUTSMF79

Diagram 98. Address Space Resource Data Gatherer (ERBGARD0) (Part 5 of 8)

No diagram

Extended Description continued on next page.

Extended Description	Module	Label

8 (continued)

- R792CMFX = RSCOMFX common area fixed frames
- R792PSWP = OUXBSPIN + OUXBSPOT, pages swapped (in and out)
- R792LPAI = OUXBLPAI, LPA pages in
- R792CSAI = OUXBCAPI – OUXBLPAI, CSA pages in (common pages – LPA pages)
- R792NLQF = R792PRFX – R792LSQA non-LSQA fixed frames
- R792TWSS = OUCBTWSS, target working set size for the user
- R792PIN = OUXBPIN, private area page-in count
- Call the count fixed frame interface:
  - R792PRFX count of private fixed frames
  - R792LSQA LSQA frames
  - R792FXBL number of fixed frames below 16-megabytes

Diagram 98. Address Space Resource Data Gatherer (ERBGARD0) (Part 7 of 8)

Process

9 Cancel the recovery environment.

10 Fill in the SMF Header and the Common Header.

Return to Process Control (ERBMFDPC or ERBMFBPC)

ESTAEASD ESTAE Exit

11 If the number of retries is less than 2, increase the number of retries counter, and restore the registers for retry at COMMON subroutine. → step 7

Otherwise, percolate abend.

To R/TM (Recovery/Termination)

Output

SMF Record 79

- Common Header
- Header Extension
- RMF Product Section
- Monitor II Control Section
- Relocate Block Sub Type 2
- Address Space Element 1
- •••
- Address Space Element n

Data Arrays for address spaces in the ARCD and ARCDJ reports

Extended Description	Module	Label
9 Cancel the Estae Exit	ERBGARD0	COMMON
10 Fill in the remaining SMF record field in common section:	ERBGARD0	COMMON

SMF79STY = 2, record subtype

SMF79ASL = length of a relocate block

- Set the number of data control sections to zero (SMF79DCN = 0)
- Set the length of the data control sections to zero (SMF79DCL = 0)
- Set the offset to the data section (SMF79ASS) to equal the address of SMF79 record + SMF79DCS + (SMF79DCL *SMF79DCN)

All other fields which have not been filled in by ERBGARD0, were set by Display Process Control.

ESTAEARD Estae Routine

Extended Description	Module	Label
11 Check register 0 to determine if the System Diagnostic Work Area (SDWA) was provided.	ERBGARD0	ESTAEARD

If register 0 = 12 (decimal), it was not. Therefore, get the address of the parameter list from register 2. If the SDWA was provided, the address of the parameter list is in the 1st word of the SDWA. The parameter list will point to the counter for the number of retries.

Check the counter of the number of retries. If less than 2, indicate that a retry routine should be given control.

If the number of retries is already 2, don't retry again, just percolate the abend.

ESTAE return codes in Register 15.

0 – continue termination

4 – retry at address provided

ESTAE Retry address (if a retry should be attempted) will be placed in register 0.

ERBGARD0 return codes:

0 – Data successfully gathered

4 – Syntax error in CPARM string

8 – Invalid entry code

16 – No address space which meets criteria (jobname or class/domain/status)

20 – Unable to establish ESTAE environment

24 – Syntax error detected in the DPARM text

Diagram 99. Enqueue Contention Data Gatherer (ERBGSENQ) (Part 1 of 12)

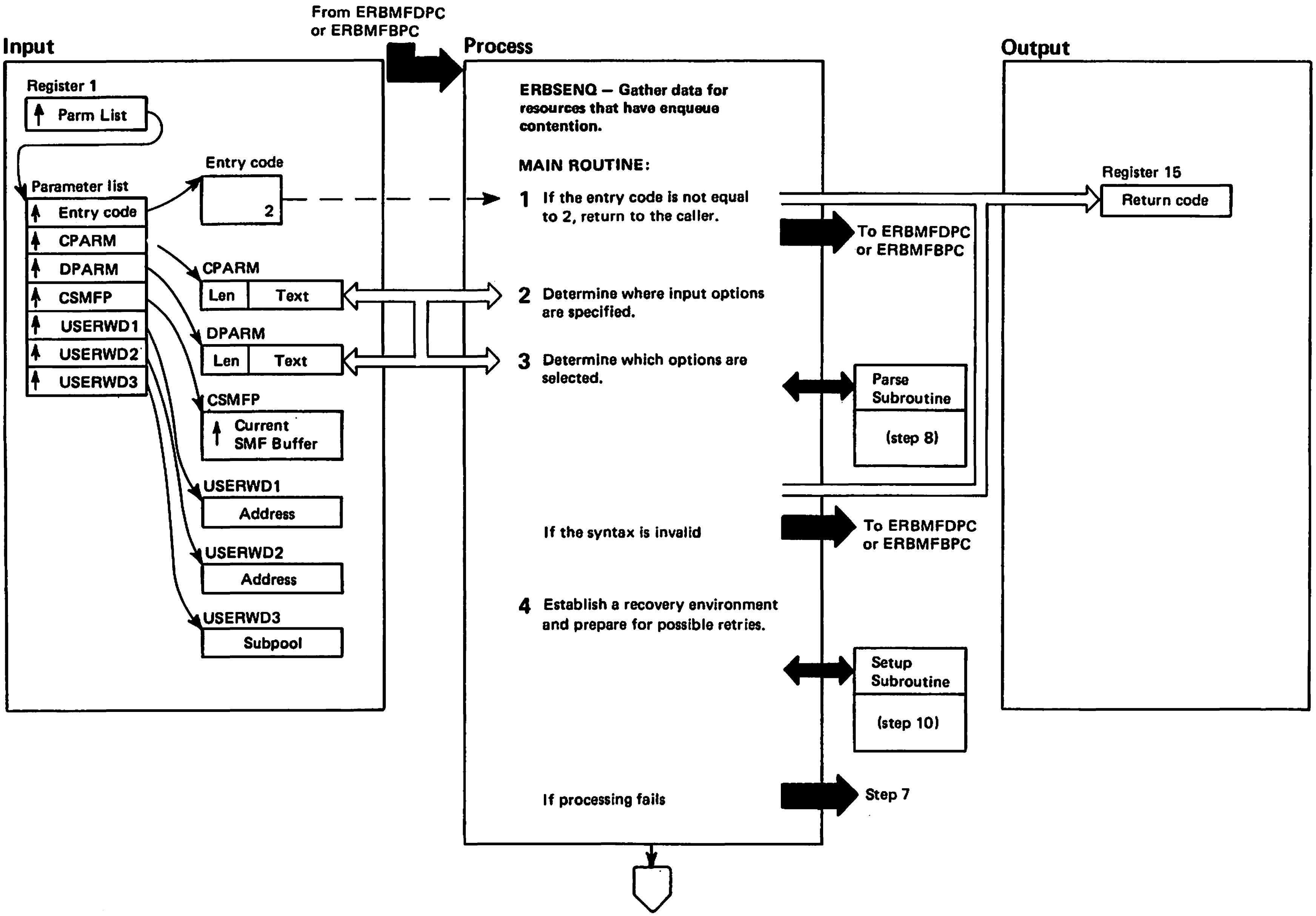
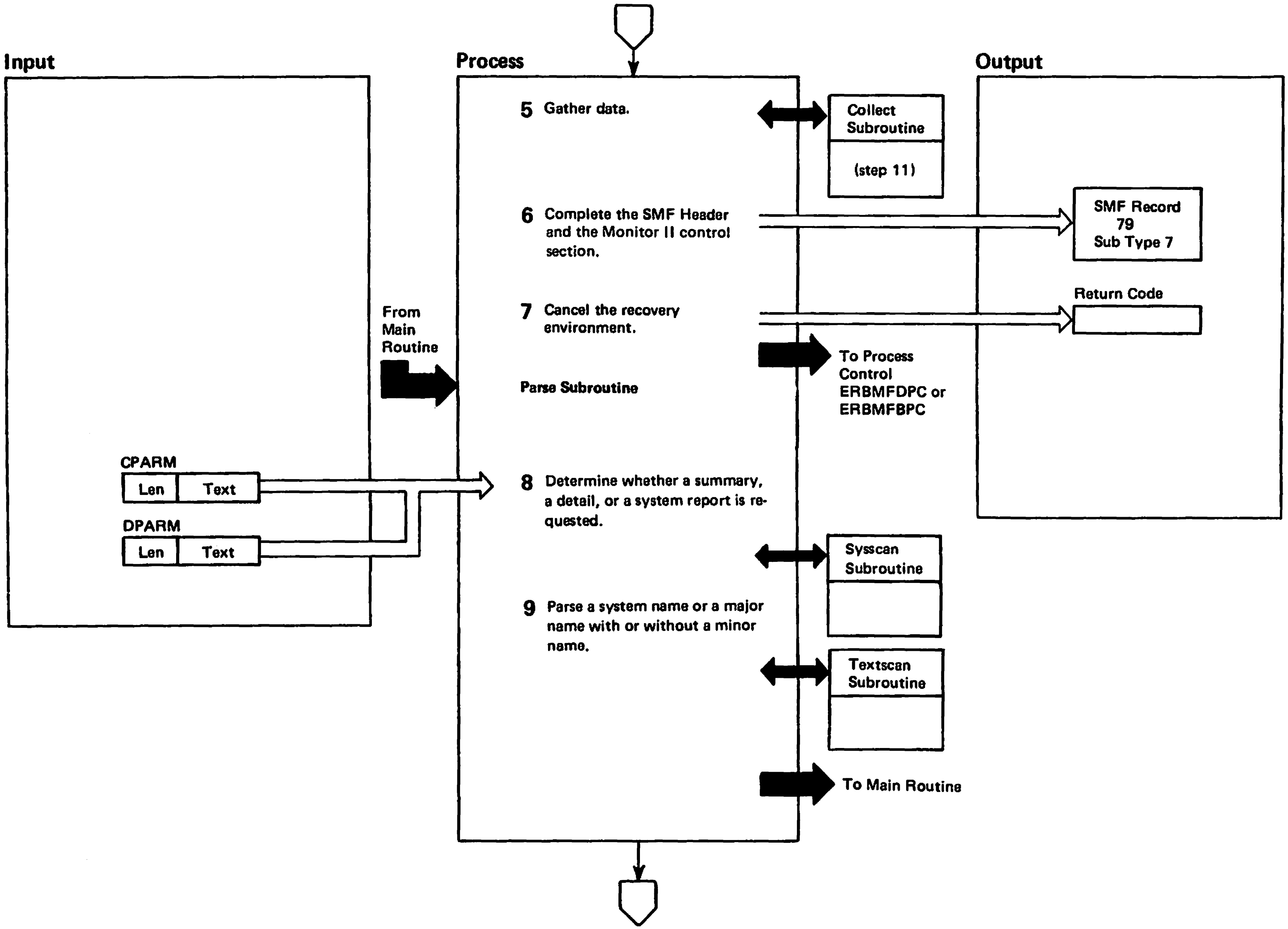


Diagram 99. Enqueue Contention Data Gatherer (ERBGSENQ) (Part 2 of 12)

Extended Description	Module	Label
The Monitor II Enqueue Contention Data Gatherer collects data for resources with outstanding enqueue contention and builds an internal image of an SMF record. (Reserve requests are not processed by this data gatherer.) Control is received from either the Display Process Control routine (ERBMFDPC) or the Background Process Control routine (ERBMFBPC).		
The variable RTCODE is used to return codes to calling routines (except in the exit routine). RTCODE is initialized to zero in the main routine before processing begins.		
ERBGSENQ ENTRY – Gather data for resources that have enqueue contention.		
1 Check the entry code. If it indicates that data should not be gathered (entry code = 2), then set a return code of 8 to indicate that no data was gathered and return to the calling routine.	ERBGSENQ	ERBGSENQ
2 Determine where the input options are specified.	ERBGSENQ	ERBGSENQ
If the CPARM length field contains a non-zero value, then the user has supplied the operands. Set PARMPTR= ADDR(CPARM) to indicate that the text to parse is in the CPARM text.		
If the CPARM length field is zero, then check to see if the DPARM length field contains a non-zero value. If the value is non-zero, then the user specified menu defaults. Set PARMPTR=ADDR(DPARM) to indicate that the text to parse is in the DPARM text.		
If the DPARM and the CPARM length field values both are zero, then set an internal default for a detail report: R797GDET=ON.		

Extended Description	Module	Label
3 If the text to be parsed is in the CPARM or DPARM text field, then call the parse subroutine to determine which report was requested.	ERBGSENQ	PARSE
If an error occurred in the parsing process, return to the calling routine.		
4 Call the setup subroutine to establish the ESTAE environment.	ERBGSENQ	SETUP

Diagram 99. Enqueue Contention Data Gatherer (ERBGSENQ) (Part 3 of 12)



Extended Description	Module	Label
5 If the ESTAE environment was successfully established, call the collect subroutine to scan and gather data on enqueue contention from GQSCAN (a service that global resource serialization provides to enable programs to determine the ENQ status of resources).	ERBGSENQ	COLLECT
6 On completion of gathering data, fill in the remaining SMF record fields in the Common Section: SMF79STY=7, record subtype for Monitor II Enqueue Contention report All fields that have not been filled in by ERBGSENQ were set by Picture Build (ERBPCTBL) and the ERBBDATA subroutine of Background Process Control (ERBMFBPC).	ERBGSENQ	ERBGSENQ
7 Cancel the ESTAE EXIT by issuing the ESTAE macro with an exit address of zero. Return to the caller.	ERBGSENQ	ERBGSENQ

Return Codes

0 — data gathered successfully
4 — syntax error in CPARM string
8 — invalid entry code
16 — no enqueue contention data was found
20 — unable to establish ESTAE environment
24 — syntax error in DPARM string

Extended Description	Module	Label
Parse Subroutine		
8 Scan the text until the first non-blank character is encountered. Parse at least the first two bytes of the appropriate text to determine the type of report requested. Check for the following syntax:	ERBGSENQ	PARSE

Text	*Flag to Set*	*Meaning*
'S' followed by a blank or end of buffer	R797GDET=OFF	The user requested a summary report.
'D' followed by a blank or end of buffer	R797GDET=ON	The user requested a detail report.
'A,' followed by a system name	R797GDET=ON	The user requested a detail report of all resources owned by a system.
'E,' followed by a system name	R797GDET=ON	The user requested a detail report of all resources exclusively owned by a system.
All other characters	R797GDET=ON R797GMAJ=ON	Any other characters are considered to be a major name with a possible minor name.
The flags R797GMAJ and R797GMIN are initialized to OFF before the actual parsing takes place.		

Extended Description	Module	Label
9 Check the R797GDET and R797GMAJ flags. If both flag bits were set to ON, then call the TEXTSCAN subroutine to parse the major name, with or without a minor name, specified in the text.	ERBGSENQ	PARSE TEXTSCAN
If a report by a system was requested (A or E), call the SYSSCAN subroutine to parse the system name.		SYSSCAN
Return to the caller.		

Diagram 99. Enqueue Contention Data Gatherer (ERBGSENQ) (Part 5 of 12)

Input

From Main Routine

Process

Setup Subroutine

10 Prepare the ESTAE exit parameter list.

Issue the ESTAE macro.

If the ESTAE macro fails, set a proper return code.

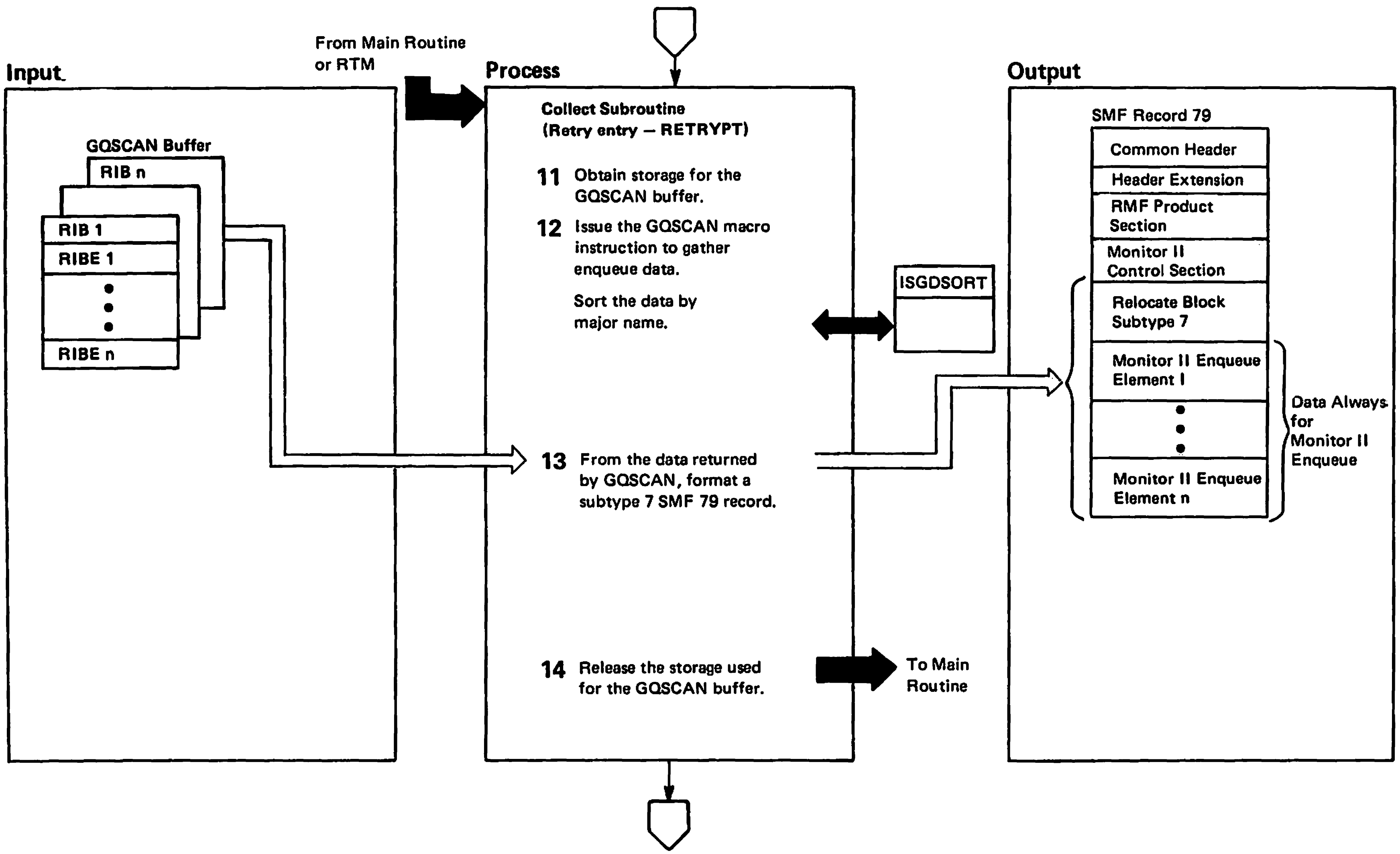
To Main Routine

Output

Return Code

Extended Description	Module	Label
Setup Subroutine		
10 Prepare a parameter list with:	ERBGSENQ	SETUP
• A counter of the number of retries		
• The address of a retry entry point		
• The save area for registers 0 through 15		
Initialize the counter of retries to 0 and the retry address to the address of the RETRYPT entry point.		
Issue the ESTAE macro to set up the ESTAE exit (ESTAERCS).		
The ESTAE exit will cover any error in collecting the contention data.		
Set a return code of 20 if the ESTAE macro fails.		
Return to the caller.		

Diagram 99. Enqueue Contention Data Gatherer (ERBGSENQ) (Part 7 of 12)



Extended Description	Module	Label
Collect Subroutine (Retry Entry – RETRYPT)		
11 If the number of retries is zero, then it is the first entry to this subroutine. Therefore, store the registers in the parameter list save area.		
If an abend has occurred and this is a retry, then load the registers from the parameter list save area and increase the count of the number of retries by one.		
Gather the enqueue contention data that is required by the user: a) Get the current time of day in decimal (HHMMSSTH) by issuing the TIME macro. Save the time in R79GTOD in the form 0HHMMSSF (hours, minutes, seconds, sign). b) Initialize the number of relocate block elements to zero (SMF79ASN=0). Set the size of each element (SMF79ASL=length) (R797ELEM). Issue the GETMAIN macro instruction to obtain storage for the GQSCAN buffer.	ERBGSENQ	COLLECT
12 Issue the GQSCAN macro instruction. GQSCAN places the resource contention data in the buffer. It creates a RIB for each resource in contention and a RIBE for each owner and waiter.		
Call ISGDSORT to chain the RIBs in the buffer in order of major name. For more information, see *Global Resource Serialization Logic.*	IDSGSORT	

Extended Description	Module	Label
13 If a specific detail report is requested (R797GDET=ON, R797GMAJ=ON, and R797GMIN=ON or OFF), scan the RIBs in the buffer, and gather information for the resource specified, if it can be found. Gather the following information from the data for each RIBE that is in contention for a particular resource; place the data for each resource in a relocate block of the current SMF buffer. • R797MAJ, major enqueue name, obtained from RIBQNAME. • R797MIN, minor enqueue name, obtained from RIBRNAME. • R797MINL, minor name length, obtained from RIBRNMLN. • R797REQ, type and status of a request for a resource. It is determined by the RIBETYPE and RIBESTAT bits. R797REQ can be one of the following: EO, exclusive request and owner of the resource EW, exclusive request and waiting for the resource SO, shared request and owner of the resource SW, shared request and waiting for the resource • R797SCOP, scope of the resource, determined from the RIBSCOPE bit settings. R797SCOP can be one of the following: SYS, scope of system; the resource used by programs of more than one address space SYSS, scope of systems; the resource used by programs of more than one address space but considered to be a different resource from SYS. STEP, scope of step; the resource used only within the job step of the requesting program. • R797ASID, address space identifier, obtained from RIBEASID. • R797JBN, name of the job in which the enqueue request was issued, obtained from RIBEJBNM. • R797SID, system name, obtained from RIBESYSN. • R797OWN, count of the number of owners of a particular resource, from RIBNTO.		

Diagram 99. Enqueue Contention Data Gatherer (ERBGSENQ) (Part 9 of 12)

No diagram.

Extended Description continued on next page.

Extended Description	Module	Label

13 (continued)

- R797EXCW, count of the number of waiters for exclusive use of the resource, from RIBNTWE.
- R797SHRW, count of the number of waiters for shared use of the resource, from RIBNTWS.

Note: Although the R797OWN, R797EXCW, and R797SHRW fields are used only in a summary report, they are filled in during the data gathering process. These fields can be used to summarize any detail data that has been gathered.

The total owning, exclusive waiting, and shared waiting counts for a resource are stored in the first element built for that resource. By storing the total counts in the first element, the Enqueue Data Reporter (ERBRSENQ) is able to suppress lines for a summary report.

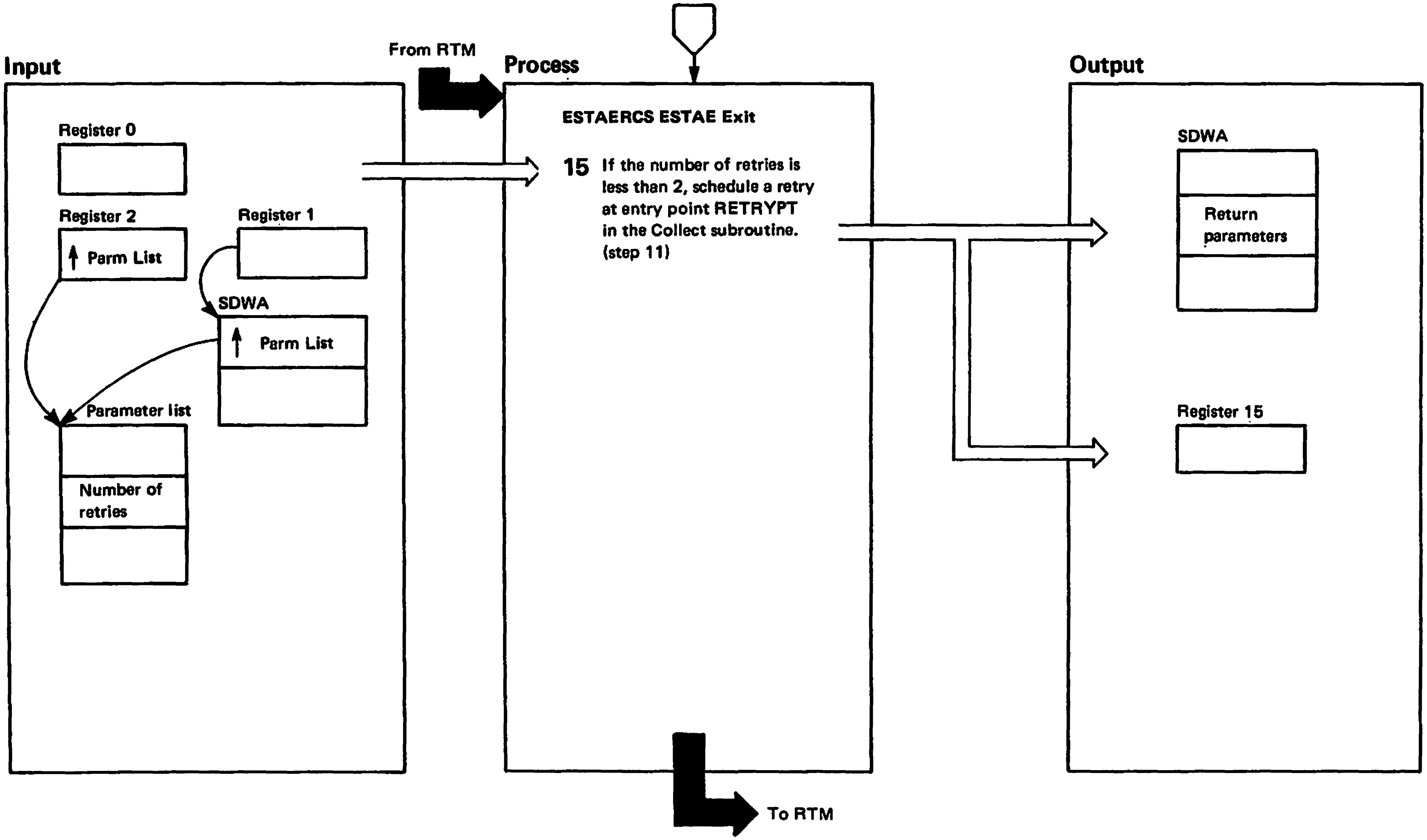
Set the R79PAR bit to ON (=1) if gathering requires more relocate blocks than there are available and end data gathering by returning to the caller. The result is a report with partial data collected.

If no enqueue contention was found for any type of report being processed, set a return code of 16.

14 Issue the FREEMAIN macro instruction to release the storage used for the GQSCAN buffer.

Return to caller.

Diagram 99. Enqueue Contention Data Gatherer (ERBGSENQ) (Part 11 of 12)



Extended Description	Module	Label
ESTAERCS Exit Routine		
15 Check register 0 to determine if a system diagnostic work area (SDWA) was provided. If register 0=12 (decimal), then it was not. Therefore, obtain the address of the parameter list from register 2. If an SDWA was provided, the address of the parameter list is in the first word of the SDWA. The parameter list has the counter for the number of retries.	ERBGSENQ	ESTAERCS

Check the counter of the number of retries. If it is less than 2, indicate that the retry routine should be given control.

- If an SDWA was available, fill in the SDWA with the retry address and a return code of 4.
- If an SDWA was not available, set register 0 to the retry address and pass back a return code of 4 in register 15.

If the number of retries is already 2, do not retry again; just percolate the abend.

- If an SDWA was available, fil in the SDWA with a return code of 0.
- If an SDWA was not available, return with a code of 0 in register 15.

ESTAE Exit Routine return codes in register 15:

0 — continue termination
4 — retry at address provided

Diagram 100. Reserve Data Gatherer (ERBGENQR) (Part 1 of 8)

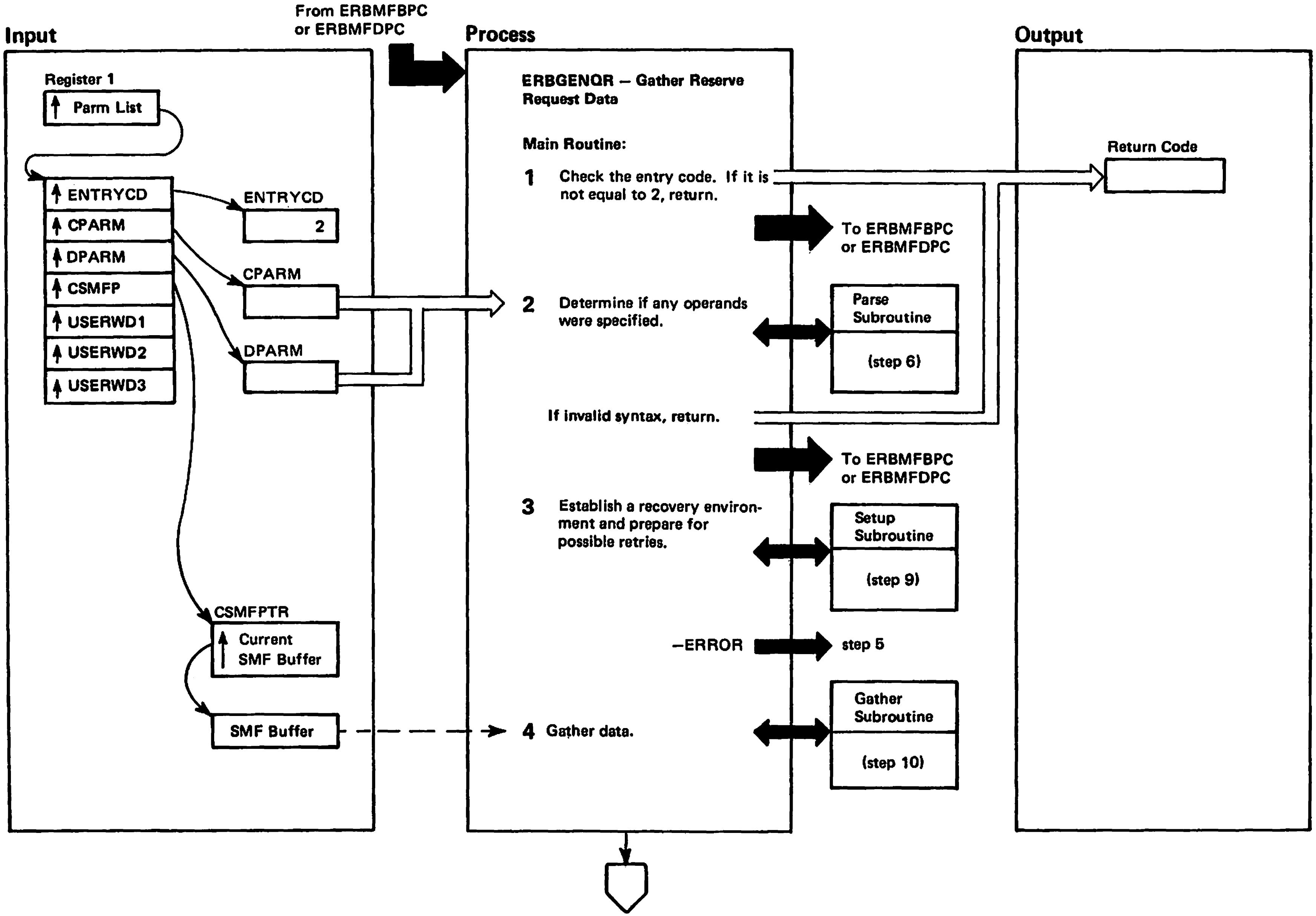
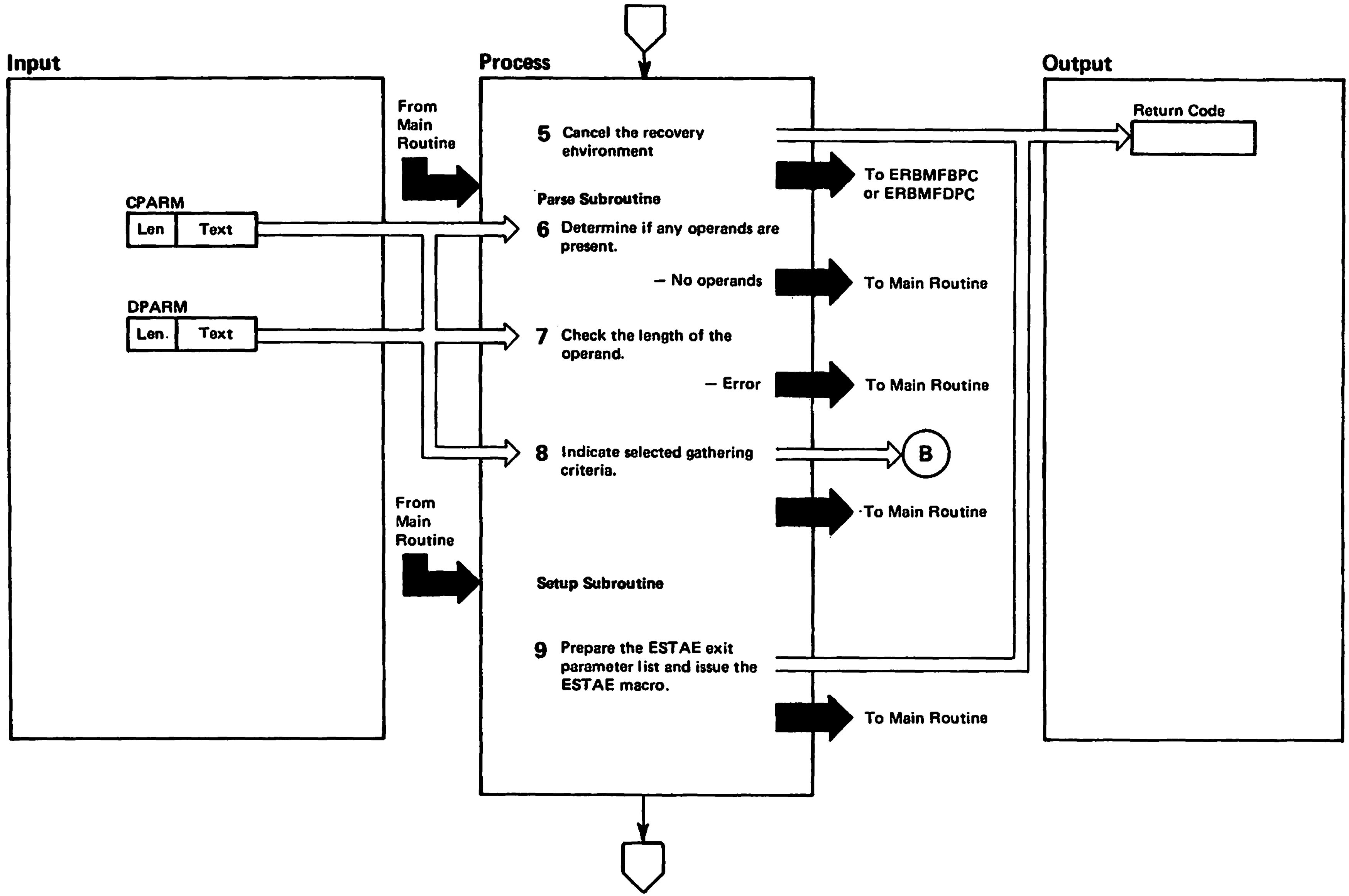


Diagram 100. Reserve Data Gatherer (ERBGENQR) (Part 2 of 8)

Extended Description	Module	Label
The Monitor II Reserve Data Gatherer collects data on resources for which reserve requests have been issued and builds an internal image of an SMF record describing only reserve request information. Control is received from either the Display Process Control Routine (ERBMFDPC) or the Background Process Control Routine (ERBMFBPC).		
The variable RTCODE is used to return codes to calling routines (except in the exit routine). RTCODE is initialized to zero in the main routine before processing begins.		
ERBGENQR ENTRY — Gather all reserve request data		
1 Check the entry code. If it indicates that data should not be gathered (entry code = 2), then set a return code of 8 to indicate no data was gathered and return to the calling routine.	ERBGENQR	ERBGENQR
2 Call the Parse subroutine to determine if an operand was specified as input. If a syntax error was encountered (RTCODE is not zero) return to the caller.	ERBGENQR	PARSE
3 Call the setup subroutine to establish the ESTAE environment.	ERBGENQR	SETUP
4 If the recovery environment was successfully established (RTCODE=0), call the Gather subroutine to gather reserve data.	ERBGENQR	GATHER

Diagram 100. Reserve Data Gatherer (ERBGENQR) (Part 3 of 8)



Extended Description	Module	Label
5 Cancel the ESTAE exit by issuing the ESTAE macro with an exit address of zero.	ERBGENQR	ERBGENQR
Return to caller.		
Return Codes 0 — data gathered successfully 4 — syntax error in CPARM string 8 — invalid entry code 16 — no data found 20 — unable to establish ESTAE EXIT 24 — syntax error in DPARM string		
Parse Subroutine		
6 If both the CPARM and the DPARM length fields contain a zero, then the user did not specify any operands as input. As a default, gather all reserve request data.	ERBGENQR	PARSE
Return to the caller.		
7 Check for an invalid operand length in the appropriate text field.	ERBGENQR	PARSE
All input requests longer than 6 characters are invalid except the operand ALLVSER (7 characters), which requests that all reserve request data be gathered.		
Operands of 1-6 characters represent a specific volume serial number and request that information be gathered only on reserve requests made to that volume.		
If the operand is found to be invalid, set a return code of 4 (CPARM text syntax error) or 24 (DPARM syntax error) and return to the caller.		
8 For later use by the Gather subroutine, set an internal field to indicate what operand was specified.	ERBGENQR	PARSE
Return to the caller.		
Setup Subroutine		
9 Prepare a parameter list with: • A counter of the number of retries • The address of the retry entry point • The save area for registers 0 through 15	ERBGENQR	SETUP
Initialize the counter of retries to 0 and the retry address to the address of the RETRYPT entry point.		
Issue the ESTAE macro to set up the ESTAE exit (ESTAERCS).		
The ESTAE exit will help recover from any error in processing the address space vector table (ASVT) chain of address space control blocks (ASCBs).		
Set a return code of 20 if the ESTAE macro failed.		
Return to the caller.		
Gather Subroutine — (Retry Entry — RETRYPT)		
If the number of retries is zero, then it is the first entry to this subroutine. In this case, store the registers in the parameter list save area.		
If an abend has occurred and this is a retry (entry at RETRYPT), then load the registers from the parameter list save area and increase the count of the number of retries by one.		

Diagram 100. Reserve Data Gatherer (ERBGENQR) (Part 5 of 8)

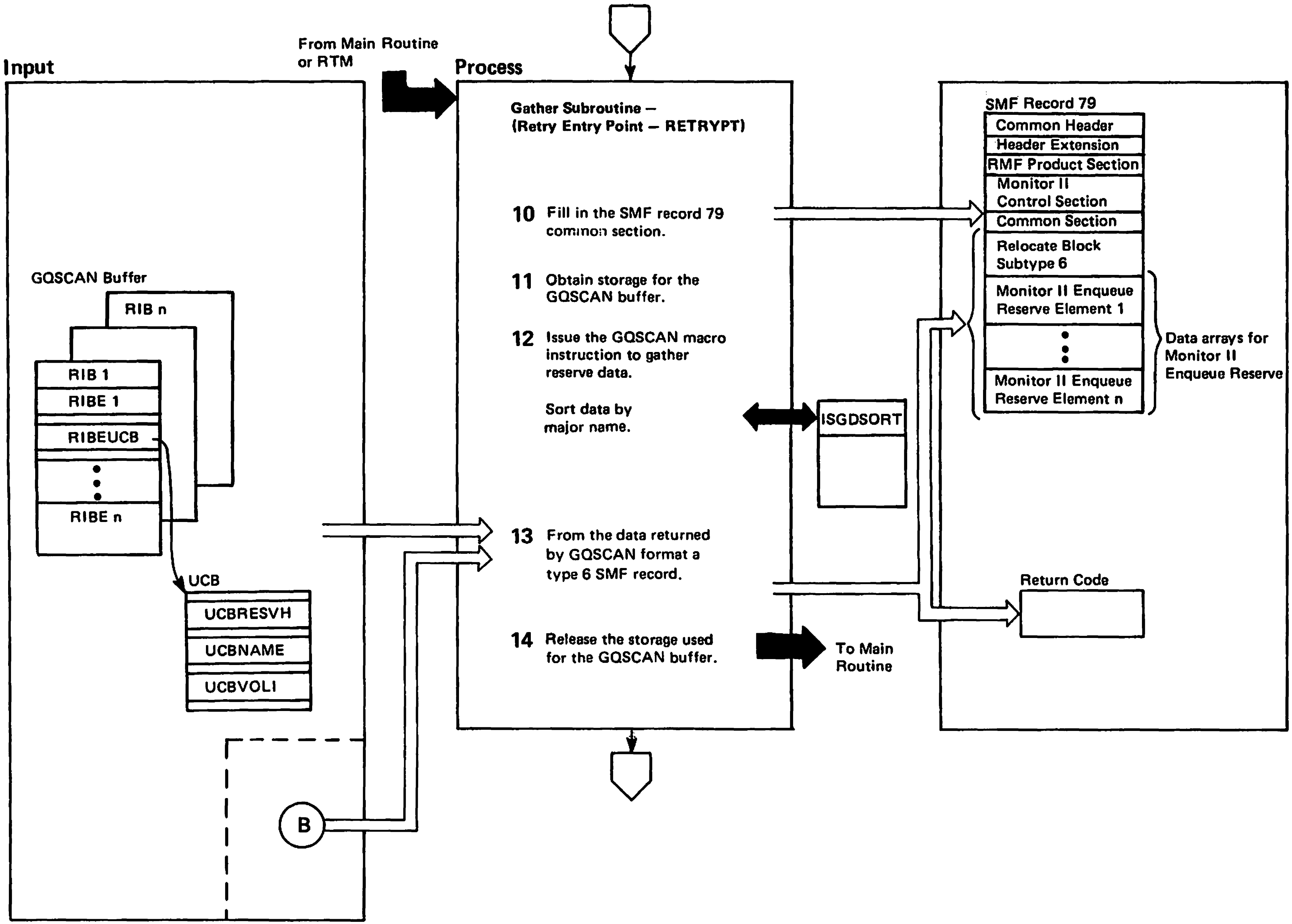


Diagram 100. Reserve Data Gatherer (ERBGENQR) (Part 6 of 8)

Extended Description	Module	Label

Gather Subroutine

10 Fill in the following fields in the common section of the SMF79 record: — Module: ERBGENQR — Label: GATHER

SMF79ASL=LENGTH(R796ELEM), size of a data element

SMF79STY=6, record subtype for Monitor II Enqueue Reserve Report

Initialize the number of data elements to zero (SMF79ASN=0). This value is increased by one each time a RIB representing a reserve request is encountered and data is collected.

11 Obtain the current ime of day by issuing the TIME SVC (HHMMSSTH format is obtained). Save the time in the R796TOD field of the SMF record in the form 0HHMMSSF (hours, minutes, seconds, sign). Issue the GETMAIN macro instruction to obtain storage for the GQSCAN buffer. — Module: ERBGENQR — Label: GATHER

12 Issue the GQSCAN macro instruction. The parameters specified on the GQSCAN macro instruction indicate that GQSCAN is to return data on all reserve resources, whether or not contention is occurring. GQSCAN places the reserve data in the buffer. It creates a RIB for each reserve resource and a RIBE for each owner and waiter.

Call ISGDSORT to chain the RIBs in the buffer in order of major name. For more information, see *Global Resourse Serialization Logic.* — Module: ISGDSORT

13 If a specific volume serial number was specified as input, then gather the following information on each RIB that represents a reserve request for that volume. Otherwise, gather the data on all RIBs. — Module: ERBGENQR — Label: GATHER

- R796REQ, type and status of a request for a resource, is determined from RIBETYPE and RIBESTAT

  R796REQ can be one of the following:
  - EO, exclusive request and owner of the resource
  - EW, exclusive request and is waiting for the resource
  - SO, shared request and is owner of the resource
  - SW, shared request and is waiting for the resource
- R796MAJ, major enqueue name, from RIBQNAME
- R796MIN, minor enqueue name from RIBRNAME
- R796MINL, minor name length, is obtained from RIBRNMLN or is equal to LENGTH(R796MIN) if RIBRNMLN > LENGTH(R796MIN)
- R796ASID, address space identifier, obtained from RIBEASID
- R796JBN, name of the job in which the reserve request was issued, is obtained from RIBEJBNM
- R796SID, system name, is obtained from RIBESID
- R796UCB, device address, obtained from the UCBNAME field in the UCB pointed to by RIBEUCB
- R796VOLS, device volume serial number obtained from the UCBVOLI field in the UCB
- R796RESV, hardware reserve status which is determined by the UCBRESVH field value in the appropriate UCB

  R796RESV can be one of the following:

  1 (=ON), device is reserved by this system

  0 (=OFF), device is not reserved by this system
- R796TRUN=y, minor name truncation indicator

  R796TRUN can be one of the following:

  1 (=ON), minor name is longer than the length allotted in the SMF record — truncation has occurred (i.e., RIBRNMLN > LENGTH(R796MIN))

  0 (=OFF), QCB minor name has not been truncated RIBRNMLN < =LENGTH(R796MIN))

Set R79PAR to ON if gathering requires more relocate blocks than there are available. End gathering data by returning to the caller. The result is a report with partial data collected.

All fields that were not filled in by ERBGENQR are set by Picture Build (ERBPCTBL) and ERBDATA subroutine of Background Process Control (ERBMFBPC) if this is a background session.

If no reserve request data was gathered, set a return.

14 Issue the FREEMAIN macro instruction to release the storage used for the GQSCAN buffer.

Return to the caller.

Diagram 100. Reserve Data Gatherer (ERBGENQR) (Part 7 of 8)

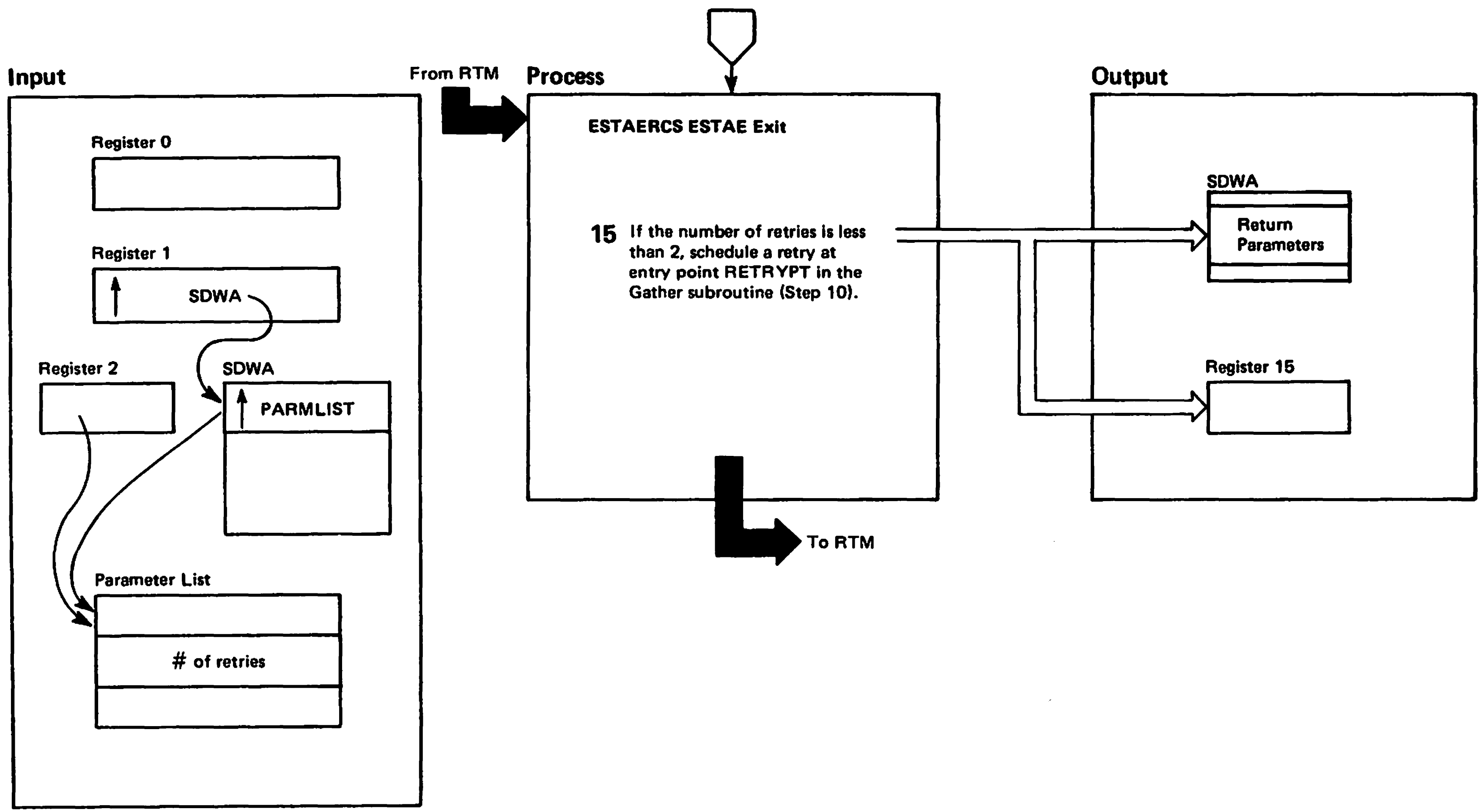


Diagram 100. Reserve Data Gatherer (ERBGENQR) (Part 8 of 8)

Extended Description	Module	Label
ESTAERCS EXIT Routine		
15 Check register 0 to determine if the System Diagnostic Work Area (SDWA) was provided. If register 0=12 (decimal) then it was not. Therefore, obtain the address of the parameter list from register 2. If the SDWA was provided, the address of the parameter list is in the first word of the SDWA. The parameter list has the counter for the number of retries.	ERBGENQR	ESTAERCS

Check the counter of the number of retries. If it is less than 2, indicate that a retry routine should be given control.

- If SDWA was available, fill in the SDWA with the retry address and a return code of 4.
- If the SDWA was not available, set register 0 to the retry address and pass back a return code of 4 in register 15.

If the number of retries is already 2, do not retry again, just percolate the abend.

- If the SDWA was available, fill in the SDWA with a return code of 0.
- If the SDWA was not available, return with a code of 0 in register 15.

ESTAE exit routine return codes in register 15.

0 — continue termination
4 — retry at address provided

Diagram 101. Real Storage/CPU/SRM Data Gatherer (ERBGRCS0) (Part 1 of 6)

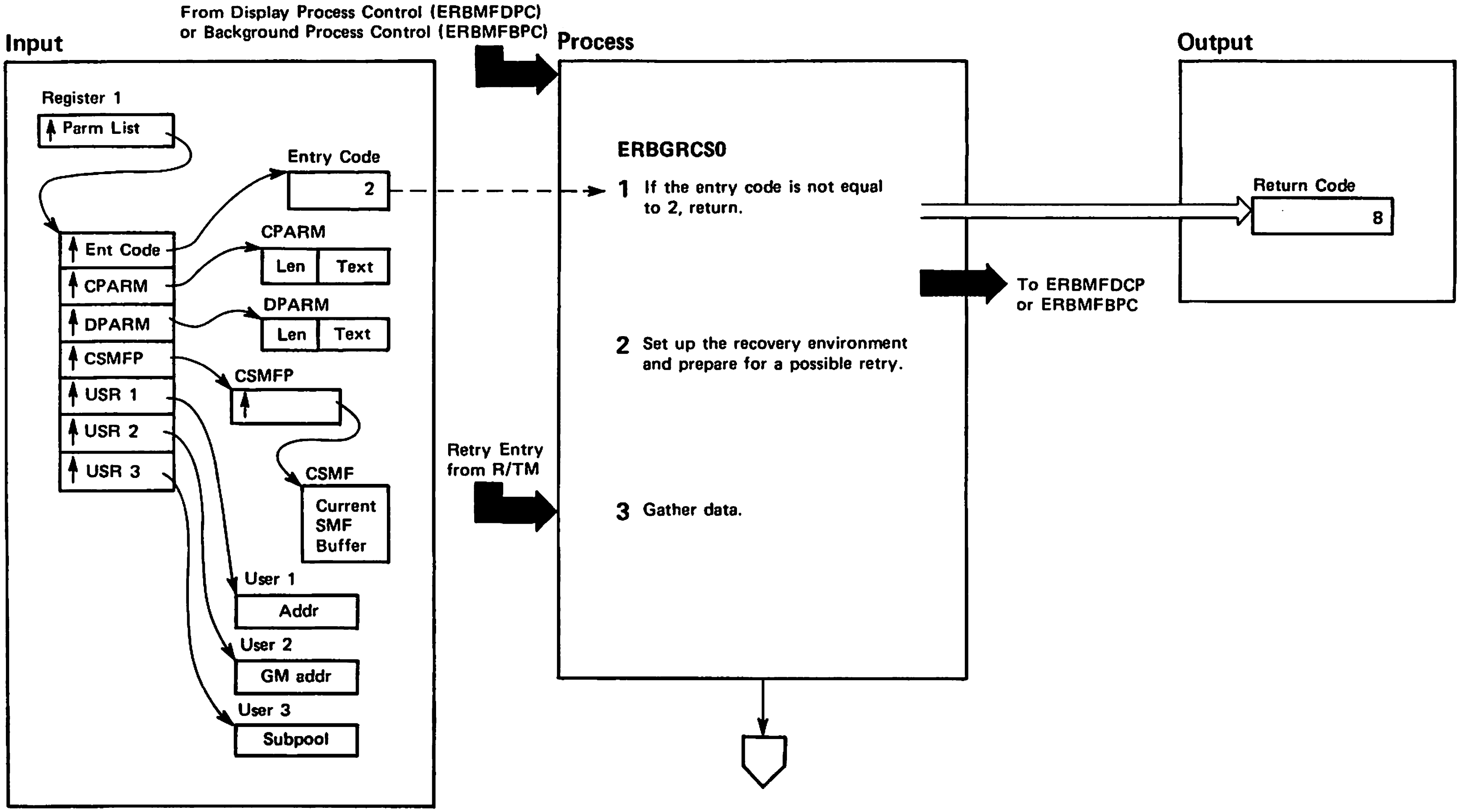


Diagram 101. Real Storage/CPU/SRM Data Gatherer (ERBGRCS0) (Part 2 of 6)

Extended Description	Module	Label
The Real Storage/CPU/SRM Usage Data Gatherer collects total system data and builds an internal image of an SMF record.		
1 Check the entry code. If it is not equal to 2, which means go ahead and process, return with a code of 8.	ERBGRCS0	ERBGRC
2 Set up the Estae environment.	ERBGRCS0	ERBGRCS0

Prepare a parameter list with:

- A counter of the number of retries
- The address of the retry entry point
- A save area for registers 0 through 15

Initialize the number of retries to 0, and the retry address to the address of RETRYPT.

Issue the ESTAE macro to define the Estae Exit (ESTAERCS).

The Estae Exit covers any errors that occur while scanning the SRM queues or the ASCB ready queue by retrying at entry point RETRYPT.

GATHER Subroutine – Retry Entry (RETRYPT)

Extended Description	Module	Label
3 If the number of retries is zero, it is the first entry to this subroutine. Therefore, store the registers in the save area in the parameter list.	ERBGRCS0	GATHER

If the number of retries is greater than zero, it is not the initial entry. An abend has occurred and the retry was scheduled. Therefore, load the registers from the save area in the parameter list and increase the counter of the number of retries by one.

Gather the data that is required by the user:

- Get the current time of day via the TIME macro in decimal (HHMMSSTH). Save it in R79GTOD in the form 0HHMMSSF (hours, minutes, seconds, sign).
- Collect the following information and save it in the current SMF buffer.
  - R793AFC=RECAFC, the count of available real storage frames (unused)
  - R793CRI=MCVSTCRI, high use count
  - R793SQA=RCESQAAL, number of frames currently allocated to SQA
  - R793CMNF=RCECOMAL, number of frames allocated to common
  - R793CMFF=RCECOMFX, number of common area pages that are fixed
  - R793PRFX=RCETOTFX–RCESQAAL–RCECOMFX, count of private area (including LSQA) fixed frames
  - R793CPUU=CCVUTILP, system CPU utilization
  - R793ASMQ=RCVASMQA, SRM measure of ASM queue length
  - R793LPAF=RCELPAAL, number of frames allocated to PLPA and PLPA directory
  - R793CSAF=RCECOMAL–RCELPAAL, CSA frame count
  - R793LPFX=RCELPAFX, number of PLPA and PLPA directory pages that are page fixed
  - R793CSFX=RCECOMFX–RCELPAFX, CSA fixed frames
  - R793LSQA=RCELSAAL, number of frames currently allocated to LSQA for all address spaces
  - R793NLQF=R793PRFX–R793LSQA, non-LSA private fixed frames
  - R793LOUT=number of address spaces logically swapped out
- Determine the length of the Address Space Control Block (ASCB) ready queue.

  The communications vector table (CVT) field CVTASCBH points to the highest priority ASCB on the ASCB dispatching queue. Starting with this ASCB, follow down the chain of ASCB s by using the forward pointer (ASCBFWDP) field which contains the address of the next ASCB on the ASCB ready queue. Stop searching the chain when the 'next ASCB address' is zero or when the number of ASCBs exceeds the ASVTMAXU count in the Address Space Vector Table (ASVT).

  Note: This second check prevents endless looping if the chain changes during the search.

  Save the final count of ASCBs on the ready queue in R793DQ.
- Determine the length of the SRM In-Queue as follows:

  Locate the System Resources Manager Queue Header Block (RMQH) via the System Resources Manager Control Table (RMCT) field RMCTINQE. The RMQH points to the first Resources Manager User Control Block (OUCB). Follow the chain of OUCBs on the queue via the OUCBFWD pointer field. Stop searching when the OUCBNAME field is not equal to 'OUCB', that is, when the last OUCB points back to the RMQH.

  Save the final count of OUCBs on the in queue in R793INC.

Diagram 101. Real Storage/CPU/SRM Data Gatherer (ERBGRCS0) (Part 3 of 6)

No diagram.

Extended Description continued on next page.